AN EXAMINATION OF MULTI-TIER DESIGNS
FOR LEGACY DATA ACCESS

THESIS
Michael L. Acker
Captain, USAF

AFIT/GCS/ENG/97D-01

19980121 071

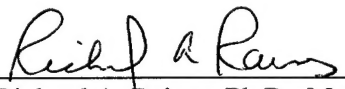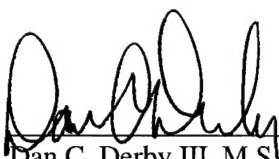Approved for public release; distribution unlimited

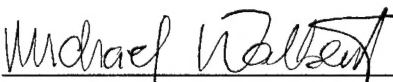AN EXAMINATION OF MULTI-TIER DESIGNS FOR LEGACY DATA ACCESS

THESIS

Michael L. Acker
Captain, USAF

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Systems

Richard A. Raines, Ph.D., Major
Committee Member

Dan C. Derby III, M.S., Captain
Committee Member

Michael L. Talbert, Ph.D., Major
Committee Chairman

The views expressed in this thesis are those of the author and do not reflect the official policy or

position of the Department of Defense or the U.S. Government

AFIT/GCS/ENG/97D-01

AN EXAMINATION OF MULTI-TIER DESIGNS FOR LEGACY DATA ACCESS

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Systems

Michael L. Acker, B. S.

Captain, USAF

December 1997

Acknowledgments

I want to thank my thesis advisor, Major Talbert, for his support and guidance throughout the thesis process. Also, my thanks go to my other committee members, Major Raines and Captain Derby, for additional support for my work at AFIT.

My greatest and most heart-felt appreciation and thanks go to my wife, Vicki, for her constant support and understanding. She managed to keep our family close and strong despite the long study hours and obvious stress and frustration of the "AFIT experience." Likewise, my thanks go to our children, Pam, Bob, Debra, and Matthew, for understanding (as best they could) this latest set of demands the Air Force was putting on Dad and family.

Table of Contents

Table of Figures

List of Tables

AFIT/GCS/ENG/97D-01

Abstract

This work examines the application of Java and the Common Object Request Broker Architecture (CORBA) to support access to remote databases via the Internet. The research applies these software technologies to assist an Air Force distance learning provider in improving the capabilities of its World Wide Web-based correspondence system.

An analysis of the distance learning provider's operation revealed a strong dependency on a non-collocated legacy relational database. This dependency limits the distance learning provider's future web-based capabilities. A recommendation to improve operation by data replication is proposed, and the implementation details are provided for two alternative test systems that support data replication between heterogeneous relational database management systems. The first test system incorporates a two-tier architecture design using Java, and the second system employs a three-tier architecture design using Java and CORBA.

Data on replication times for the two-tier and three-tier designs are presented, revealing a greater performance consistency from the three-tier design over the two-tier design for varying client platforms and communications channels. Discussion of a small-scale proof-of-concept system based on the three-tier design is provided, along with a presentation of the potential for the technologies applied in this system to benefit Air Force web-based distance learning.

An Examination of Multi-tier Designs
for Legacy Data Access

I. Introduction

This research examines the application of two recently introduced technologies, Java™ and CORBA™, to support access to remote databases via the Internet[1]. The research is performed in the context of assisting an Air Force distance learning provider in improving the capabilities of its World Wide Web-based correspondence system.

1.1 Background

The Air Command and Staff College (ACSC) is the intermediate professional military education school for the Air Force. ACSC is responsible for "educating midcareer officers to lead in developing, advancing, and applying air and space power across the spectrum of service, joint, and combined military operations" [ACSC WWW97]. ACSC allows students to attend the in-residence version of its course at Maxwell Air Force Base in Montgomery, Alabama, while those who are unable to attend in-residence may enroll in the correspondence version of the course through the ACSC Distance Learning Program.

Since October 1995 the ACSC correspondence course has been available on CD-ROM (compact disc-read only memory) to support several features—most notably multimedia presentation and interactive environments—that are not available through book-based correspondence. As of June 1996, ACSC began providing supplementary support for the correspondence course through an Internet World Wide Web (WWW) site maintained and operated by the ACSC staff at the ACSC facilities. The ACSC WWW site provides access to presentation material and research papers while overcoming some of the distribution problems

---

[1] Java is a trademark of Sun Microsystems, Inc. CORBA is a trademark of the Object Management Group, Inc.

associated with circulating both the book and CD-ROM version of the course. The role of the WWW site has since increased with the addition of features such as an electronic bulletin board service (for student and instructor question and answer postings), a chat room (to support text-based real-time seminar-style discussions), and streaming audio (to provide audio presentation of material).

The ACSC Distance Learning Program plans to offer the entire correspondence course to new students via the WWW site in the near future. Therefore, the WWW site must incorporate significant additional capabilities beyond those already available.

Among the first additional features demanded of the WWW site is the support for a secure and easily maintainable WWW-based course registration system. Several technologies have potential for supporting the needs of the registration system requirements of the ACSC WWW-based Distance Learning Program, two of which—Java and CORBA—are the focus of this research.

1.2 Importance of Research

The ACSC Distance Learning Department has developed a strategic plan [ACSC DL97] that reflects short-term and long-term goals for providing the ACSC course. Among the many capabilities and features envisioned for the future of the ACSC Distance Learning Program is the operation of an on-line course management system. This course management system will manage the web-based distance learning program for approximately 9,000 ACSC students, providing features ranging from on-line enrollment and testing to real-time war-gaming simulation and seminar-style tele-conferencing. (See Appendix A for the complete list of proposed features.)

This thesis research will help the sponsor refine requirements for either future in-house development or acquisition efforts to support an enhanced ACSC WWW-based distance learning system. The proof-of-concept system developed at the end of this research, while being limited

in capability, also serves to demonstrate the effectiveness and capability of applying these technologies to Air Force distance learning needs.

- WWW-DBMS Integration

The rapidly growing demand for products that either support or are accessible through the Internet has affected many computing domains. Of interest to this research is the strong relationship between database management systems (DBMSs) and the WWW. The primary reason for the growing relationship between the WWW and DBMSs is the fundamental role that DBMSs serve: they provide efficient storage of and access to persistent data. In the context of DBMS and WWW integration, there are two broadly defined DBMS "customers": (1) end-users (humans and applications) requiring access to remotely located databases and (2) the web servers hosting the numerous WWW sites.

In simple terms, the Internet introduces a high degree of connectivity between non-collocated computing resources. Those resources may, in fact be geographically near or separated by vast distances. In the recent past, adding an Internet connection to an organization enabled support for e-mail, file transfer, and other basic remote computing services. With the introduction of the WWW and web browser technology, an organization gains greater productive capability through a (fairly) consistent and "friendlier" graphical user interface. As a result, organizations can now create their own *web sites* that can be open to the entire Internet community. Or an organization may operate an *intranet*, which provides service and restricts access to only those users within the organization.

One of the difficulties, however, with maintaining a "web" for an organization is the overhead of managing the traditional file-based web pages. DBMSs offer the promise of easing the burden of web site management by letting the DBMS control the pages. Plus, DBMSs offer other significant features—version control and recovery mechanisms, for example—that support web administrative duties which might otherwise be neglected.

3

Meanwhile, the importance of web administration is coming to light as policies governing federal agency web sites are being explored. The issue of maintaining web documents as federal records may lead to a requirement to archive many or all web files used on official government sites [Harreld96]. As a result, it would be prudent to take advantage of all the support features that DBMSs already provide to preserve and maintain data.

Remote database access, as depicted in Figure 1, is a goal of many commercial and non-commercial organizations investigating use of the Internet. Such support can be accomplished by incorporating (1) Internet connections at both the remote and central sites, (2) a WWW-to-DBMS connection at the server site, and (3) a general purpose web browser at the client site. The details, and issues, of how the Internet connection supports this access are discussed in Chapter 2 of this thesis.
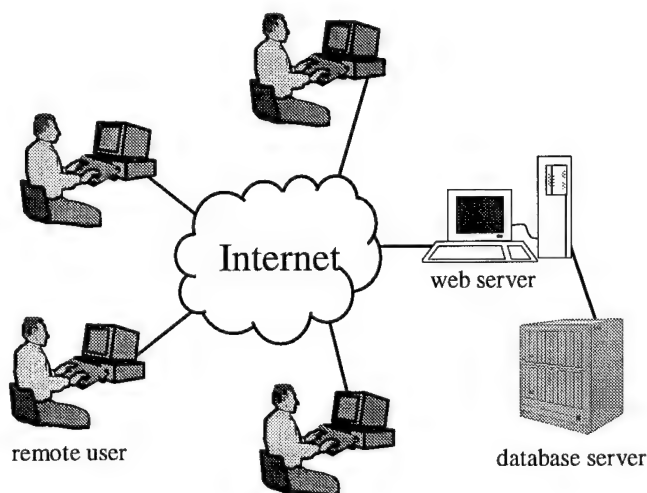
Figure 1. Remote Access to a Database Through the Internet

Another factor affirming the importance of WWW and DBMS integration is the "Air Force 2025" vision of the future of the Air Force. The Air Force 2025 doctrine [2025 Report96] includes a concept called the *adaptive learning environment* (ALE) [Sikes96], which is the expected and preferred method for training and educating the "Brilliant Warriors" comprising the Air Force 30 years from now.

A key component of the ALE model is the use of new technologies. Two of the technology areas emphasized by the ALE model are delivery systems (to transfer the training information to each airman) and tracking systems (to actively monitor each airman's career, accomplishments, and training progress and needs). The WWW, while still falling short of the advanced delivery systems envisioned by Air Force 2025, is a definite step toward meeting future Air Force training and education needs. Meanwhile, the tracking systems of the future will require some form of database management system accessible through the available delivery systems. Efforts taken now to research and develop WWW and DBMS integration directly serve the goal of developing the adaptive learning environment of the future.

1.3 Problem Statement

The ACSC web-based correspondence course has several problem areas that demand immediate attention before expansion into more complex distance learning features is possible. Two of these problem areas are:

- ACSC staff relies on a remotely located, centrally managed database for student enrollment information. Access to the student records in this database is manually-intensive and limited to a primitive command-line interface.

- Additional student contact information necessary to support the web-based course is not collected or stored by the ACSC staff. Collection, maintenance, and accessibility of this information will be crucial to effective operation of expanded versions of the web-based course.

The thrust of this thesis effort is to analyze and apply emerging network-centric and database technologies to the real-world needs of the ACSC WWW-based correspondence course. This goal is accomplished through the analysis of the current methods for working with correspondence student data at ACSC, followed by an examination of how to apply several of the recently-introduced computing technologies allowing remote access to relational database management systems. Consequently, the results of this research will be applicable to other Air

Force WWW-based distance learning providers that rely on relational database management systems.

1.4 Thesis Overview

Chapter 2 provides terminology and background information in the areas covered by this thesis, with focus on the World Wide Web, Java, CORBA, and existing WWW-based course management systems. Chapter 3 provides an analysis of the operating environment of the ACSC distance learning system and presents a recommendation for improving that operation. Chapter 4 describes the design and implementation of two approaches—using two-tier and three-tier architecture--for the system described in Chapter 3. Chapter 5 describes the performance of the two designs of Chapter 4, with a subsequent recommendation to use the three-tier design for a proof-of-concept system. Chapter 6 summarizes the conclusions and recommended areas for future research.

II. Background

This chapter provides background information on the concepts and technologies discussed in this thesis. The first section presents fundamental terminology and concepts. Subsequent sections describe the use of DBMSs to support web site management as well as several techniques for accessing existing databases through the Internet and World Wide Web. The chapter concludes with a brief review of course management systems being researched at universities or developed as commercial products.

2.1 General Concepts and Terminology

The following concepts and terminology are provided for readers not familiar with the WWW or DBMSs.

World Wide Web - (a.k.a. WWW or Web) The World Wide Web network model was developed at CERN, the European Laboratory for Particle Physics, in an effort to construct a distributed hypermedia system. The WWW model consists of interconnected computers serving information to clients. Communication between the servers and clients is accomplished using the HyperText Transfer Protocol (HTTP), and text documents that are transferred between the server and client conform to the HyperText Markup Language (HTML) (discussed below). Any object—which means more than just HTML documents—that is available on a particular *web* can be retrieved by specifying its Uniform Resource Locator (URL) (also discussed below). A *web* (note the lower case spelling) in the context of this paper refers to *any* network configuration supporting the features of the World Wide Web network model, which means such a web is not necessarily accessible by the public. The World Wide Web, in particular, refers to the specific web hosted by the many web servers connected to the Internet and whose web sites are publicly accessible through the Internet.

web server - (a.k.a. web server, HTTP server, or HTTP daemon) A web server is a specific type of software process[2] running on a computer system that maintains a web or is connected to the WWW. The web server process communicates with client software processes—typically, web browsers—using HTTP and accepts web client requests in the form of URLs. The web server processes the client requests and "serves", or transmits, the appropriate web objects back to the client. Typically, the web objects that the server retrieves are HTML documents (a.k.a., web pages or web documents) located within the file system of the host computer. The web server transmits an HTML document back to the client, which then presents the document to the user in its display area.

URL - The Uniform Resource Locator (URL) is the full address of an object on a web. Traditionally, most web resources have been static HTML pages, in which case each URL usually refers to a specific file located in a directory reachable by a web server. For example, entering the text string

http://www.afit.af.mil/Schools/EN/Default.html

into a web browser indicates a request for the file called *Default.html* from the directory *Schools/EN* within the web controlled by the web server at site *www.afit.af.mil*. The *http* prefix indicates that the remaining information in the URL conforms to hypertext transfer protocol (discussed below).

Besides the HTTP prefix, a URL may also begin with other prefixes that specify, for example, a file transfer from an *FTP*[3] site (using the "ftp:" prefix), a *Telnet* session login to a remote computer (using the "telnet:" prefix), a newsgroup to view (using the "news:" prefix), or a

---

[2] The host computer running the web server process is sometimes referred to as the web server—i.e., the hardware is called the server rather than the software. While the host computer may, in fact, be dedicated to running the web server process, often the same host computer runs other software processes—e.g., a database server process or mail server process.

[3] FTP (File Transfer Protocol) and Telnet are two widely used utilities supported over TCP/IP networks, such as the Internet. FTP supports file transfers between the server and client while Telnet provides a character-based terminal session for logging into and interacting with a remote computer.

local file to display (using the "file:" prefix). For these other URL prefixes, the information that follows the prefix conforms to the particular access method [Boutell96]. URLs may also contain extra information that is transferred to the web server, allowing it to generate HTML documents "on-the-fly" in response to user requests. This dynamic creation of web pages will be discussed more later.

web browser - A web browser (a.k.a., WWW browser) is the client software that displays HTML documents. This client software executes as a process on a computer that is connected to the network hosting the web. A person makes requests to any web server on the network by either directly entering URLs into the client software or by selecting document-embedded links—typically, hypertext links—that are themselves URLs.

HTTP - HTTP, or HyperText Transfer Protocol, is the communication language "spoken" between web clients and web servers. A key aspect to keep in mind with HTTP is that it is *stateless*, meaning no historical information (state) is stored during the time between client/server transactions. As result, of HTTP being stateless, each transaction between a client and server requires a new connection. In other words, every time the client requests a new document from the server, the server has no knowledge of who the client is or what it has requested before simply based on the HTTP information. However, this stateless nature of HTTP (and, therefore, the stateless nature of the WWW in general) can be partially overcome by password authentication methods and through the use of *cookies*[4].

web page - A web page is an HTML document that is accessible through a web. A web page may contain references to other multimedia objects—such as graphics, audio, video, or

---

[4] A *cookie* is a small file created by the web server and sent to the client for storage on the client computer. The cookie file contains state information about the client. For example, the cookie stored on the client may contain some representation of the *identity* of that particular client. When the client makes additional server requests (performs future transactions), the cookie information is sent with the URL back to the server so that the server recognizes the identity of the client and can refer to transaction history maintained by the server [Cookies96].

applets (see Section 2.3.3.3)—that will be displayed within the web page by the client web browser.

Hypertext - Hypertext refers to words or phrases within a document that can be linked to, or point to, another document [Whetzel96]. In the case of documents within a web, each *hypertext link* is translated into a URL that is used to request the next document of interest from the owning web server. Typically, embedded hypertext links are displayed as highlighted or underlined text. However, web browsers that display graphic images allow the user to select pictures or icons that represent URLs—for example, several button images may be displayed with each representing a possible user action. Or, in some cases, the web browser transmits the coordinates of the user's mouse click over an image back to the web server. The web server then looks up the associated URL for that position on the graphic and returns the appropriate web document.

HTML - HTML, or HyperText Markup Language, is the language for authoring web documents. HTML is a subset of standard generalized markup language (SGML), which attempts to standardize the formatting codes (or tags) used in text documents [Whetzel96]. An important consideration with HTML is that it only specifies how a document is structured; the language has no control over how the document is presented to the end-user. The WWW browser displays the information and is not necessarily constrained to a particular presentation standard, although most browsers conform to some version of the HTML standard[5]. This explains why WWW pages may appear differently when viewed through different browsers.

Database - According to Date, a database "consists of some collection of persistent data that is used by the application systems of some given enterprise" [Date95: 39]. In simple terms, a database preserves pieces of information for future access.

---

[5] The latest proposed version of the standard is HTML 3.2. The World Wide Web Consortium (W3C) [WC3-96] is the organization responsible for approving and maintaining the HTML standards [HTML96].

DBMS - A database management system, or DBMS, is the software that manages all access to a database. When operating in a client/server environment, the DBMS software process is often referred to as the database server. A DBMS provides many functions beyond simple access to the contents of a database. The additional features that are often expected in a DBMS are [Date95: 39-41]:

- Data definition - The DBMS must process language descriptions defining the schema (structure) of the database and create the appropriate data objects.

- Data manipulation - The DBMS must process requests to retrieve, update, or delete existing data within the database.

- Data security and integrity - The DBMS must be able to monitor user requests and reject those attempts that violate defined security or integrity rules.

- Data recovery and concurrency - The DBMS must enforce recovery (from failed transactions or system crashes) and support concurrency (shared access by multiple users).

- Data dictionary - The DBMS must provide a dictionary containing definitions about the other objects in the system—i.e., *metadata*, or information about the data.

- Performance - The DBMS should perform all of its functions efficiently.

Relational database - A relational database is "a database that is perceived by the user as a collection of normalized relations" [Date95: 98]. In basic terms, a relational database appears to the user as collection of tables containing raw data. "True" relational databases only store simple data within each relation—for example, numbers, characters, or text strings. Complex data types—such as nested records, the complex "objects" of object-oriented programming, and multimedia data (audio, video, etc.)—cannot be stored directly within the relations.

Relational DBMS - A relational DBMS is simply a DBMS (as previously defined) that manages a relational database using functions that are tailored to the relational view of the data.

11

Object-oriented (OO) database - An object-oriented database, or object database, is a database that is perceived by the user as a collection of objects conforming to the object-oriented paradigm[6].

Object-oriented DBMS - An object-oriented DBMS (OODBMS), also known as an object DBMS, is a database management system that manages an object-oriented database using functions that are tailored to the object-oriented view of the data.

Database server - A database server is a DBMS software process that manages database access in a client/server architecture [Date95: 42-43].

2.2 DBMS Support for WWW Servers

A file-based web (where each HTML document is stored as a file in a directory) can be construed as a form of database where the web server—the software program that manages HTML page requests—plays the role of a simple DBMS. Using this analogy, the database contents are the HTML text files, as well as the audio, video, graphics, applets, and other files, that make up web pages. Meanwhile, from this perspective, the relationships among the contents of this WWW database are represented by URLs.

An alternative to relying on a file-based web storage strategy is to use a true DBMS to store the components of a web. (Figure 2 emphasizes the relationship that a DBMS might have with a web server in managing different forms of data.) By using a DBMS rather than directory storage, the web site gains all the extra capabilities of a DBMS as described in Section 2.1. However, DBMS support for storing web pages and the other related data files has not been standardized, and various data models are available, each of which requires its own DBMS implementation.

---

[6] Though the object-oriented concepts among different OO proponents are similar (e.g., see [Booch91] and [Rumbaugh91] for two representations of OO methodology), object-oriented technologies are still not standardized. Standards in use or still being refined for OO support in DBMSs include SQL3 [Melton96], CORBA [OMG], and ODMG-93 [ODMG93].
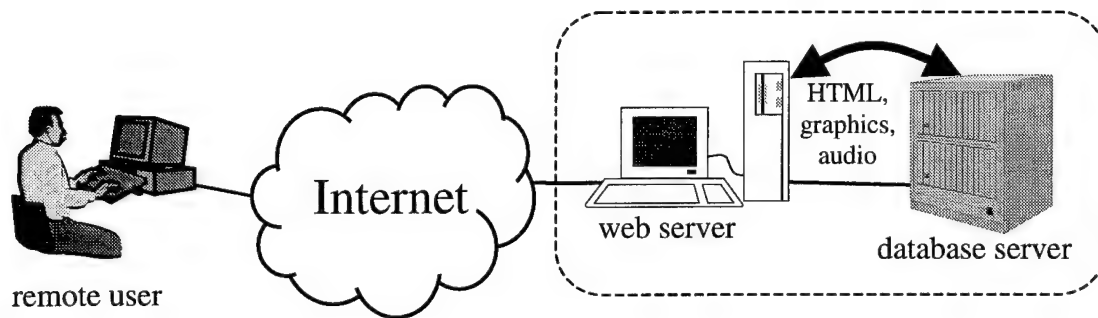
Figure 2. DBMS support for a web server.

The primary types of DBMSs available are relational DBMSs (RDBMS), object-oriented DBMSs (OODBMS), and object-relational DBMSs[7] (ORDBMS). While relational DBMS vendors continue to claim that their products are the answer to every database need, the multimedia nature of the WWW generally makes RDBMSs not well-suited for web site support. Consequently, several RDBMS vendors have developed extensions or add-ons to their DBMS products that both provide some multimedia support and allow the DBMS to act as the storage and retrieval mechanism for the many web resources. For example, RDBMS vendors like Oracle (with the Oracle 7.3 Server and Server Options [Oracle96]), Informix (with the Informix Universal Server and DataBlade modules [Aberdeen95]), and IBM (with the DB2 Universal Server and extenders [DB2-96]), can store complex, multimedia objects as binary large objects[8] (BLOBs) and offer web server applications [Montinola96] that connect to their RDBMS products directly or through the Common Gateway Interface (CGI) (discussed in Section 2.3).

Because the WWW is made up of many different types of objects that are connected via links, it has also been described as an object-oriented operating system [Merle96]. Consequently,

---

[7] Object-relational DBMSs provide some degree of support for both relational databases and object-oriented databases. ORDBMSs are appearing on the market from well-established RDBMS vendors. These RDBMS vendors are trying to meet customer demands for object-oriented support while maintaining the needed support for existing relational databases.

[8] Support for binary large objects allows these DBMSs to store the binary files—such as graphic images, audio clips, or executable programs—external to the database. The database then contains a reference to each external file—for example, by storing the file's name and location—plus, possibly, some description or keyword search information relevant to the file.

13

OODBMSs are well-suited to directly support the needs of a web site [Atwood96]. Due to this object nature, OODBMSs and object-relational DBMSs are receiving greater attention and promotion as web DBMSs. Companies such as Informix (with the Illustra Server [Baker]), ObjectStore [Objectstore96], and Oracle (with the recently released Oracle 8 Server [Bloor96]) have introduced DBMS products that claim to ease management of the many objects that comprise a web site.

2.3 Database Access Through the WWW

The alternative scenario to consider when discussing connecting databases to the WWW (or any web) is that of using the WWW to provide access to new or existing organizational databases. In Figure 3, the highlighted areas emphasize the interaction between a remote database user and a database containing organizational data. The Internet provides the communication channel, and the web server either actively supports all database use or it simply assists in establishing the connection between the user and the database server. The user, consequently, can interact with the relational data or objects being stored in the database. The following sub-sections describe some of the issues surrounding WWW access to remote databases as well as several of the current methods for providing access to databases through the WWW.
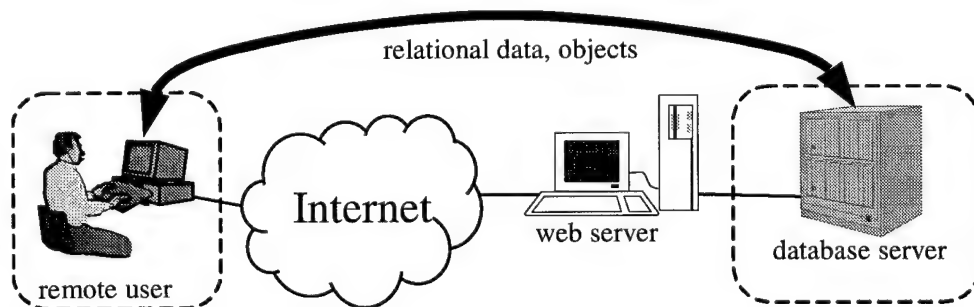
Figure 3. Interaction Between a Remote User and a Database.

14

2.3.1 Web-DBMS Issues

Several important issues should be considered when discussing database access through the WWW. These issues primarily are raised as a result of the inherent features of the WWW.

2.3.1.1 Stateless Nature of the WWW

As discussed under HTTP in Section 2.1, each transaction over the WWW between a remote client (i.e., web browser) and the web server requires a new connection. Therefore, session-oriented connections—like those established using FTP or Telnet over the Internet or between a remote database user and a database server in a client/server architecture—are not possible through the WWW or an intranet.

At first inspection, this stateless nature would seem to place a severe restriction on the use of the WWW. For example, if database access requires user authorization, and an authorized remote user wished to perform several consecutive transactions (i.e., a session of transactions) with the DBMS, then the user would have to re-enter the authorization information with each transaction. However, it is possible to simulate an extended session using the Internet "cookie" technology discussed earlier. Also, recent releases of web browsers preserve login information for sites requiring authentication, so after the first transaction the user may never know that he or she is being logged in and out with every subsequent transaction.

Other proposals have been made to enhance HTTP with ability to preserve state (so that the appearance of a session is maintained), including the use of a "State-Info" header [Kristol95]. But none of these other proposals have, as yet, been standardized.

2.3.1.2 Security on the WWW

Use of the WWW introduces additional security concerns for storing and transmitting data. These concerns are primarily associated with use of the Internet, but some apply to intranets as well. Some typical security concerns and contemporary methods to handle them are discussed next.

- Information Protection and Access Control

Data transmitted through the WWW is normally sent "in the clear" (i.e., public channel, not encrypted); therefore, anyone with access to the communication channels and who has a Web browser or suitable text viewer (or other network traffic analyzer) can intercept and read WWW traffic. Note, however, that this issue is not as great a concern for intranets (either physically isolated from the Internet or maintained behind a firewall) since the web data only travels through the organization's internal networks.

Also, unless specific restrictions are put in place, all web documents placed on the web site are accessible to everyone on the web. This unrestricted access issue applies to both the Internet—where any Internet user can retrieve the site's documents—and to intranets—where all employees with intranet access can get to all intranet web documents. In the case of a web server providing database access through its web documents, an improperly designed system could allow unauthorized users to access the organization's database.

- Methods for Access Restriction

Protection of sensitive data should always be a major concern when developing any system allowing database access. Several methods have been developed to deal with the security concerns arising from use of the WWW and, consequently, also apply to restricting access to databases connected to the WWW. Stein discusses the following three methods that restrict access to web sites and the data available through them [Stein96]:

- **Restriction by IP address, sub-net, or domain.** Using this method, the web server may be configured so that individual web documents or whole directories can only be accessed by web browsers connecting from authorized (listed) IP addresses, sub-nets, or domains. This method prevents "nosy" people from trying to access restricted data. However, determined hackers can "spoof" an IP address, making their own IP address appear to be a different address. Also, this method does not speak to the problem of unauthorized individuals retrieving restricted information by using a remote computer that has an authorized IP address.

- **Restriction by user name and password.** Using this method, the web server is configured such that a remote user must provide a name and password before

16

being allowed access to restricted web documents or directories. Normal password concerns—like password sharing or using easy-to-guess passwords—apply with this method. Also, if the data is not encrypted, the name and password could be intercepted while being transmitted to the server.

- **Encryption using public key cryptography.** Using public key encryption, both the document request and the web document, itself, are encrypted during transmission. Therefore, only the intended recipient may read the document. The two primary encryption standards are SSL (Secure Socket Layer) [SSL96] proposed by Netscape Communications Corporation and SHTTP (Secure HTTP) [SHTTP96] proposed by CommerceNet. Currently available web browsers may support both, one, or neither of these proposed standards.

2.3.1.3 Performance

Using the Internet as the communication channel for any client/server system will generally be slower than a similar system operating on an in-house network. However, the benefits of WWW access will still drive many organizations to use web technologies for database access. Therefore, to guarantee acceptable performance, an organization must analyze the many factors affecting how well the Internet, or an intranet, will perform for providing database access. Those factors that will most likely affect actual performance are as follows:

- **Bandwidth of the communication channel.** A communications channel between the client and server that includes a low-speed connection—such as a modem over the public telephone network—will likely produce unacceptable performance when transferring large amounts of data, as with audio, video, or applet files. However, low-speed connections generally show acceptable performance for text-only (or mostly text) web documents.

- **Network traffic load.** Heavy data traffic on the Internet, or even on an intranet's network, can seriously delay (or even stop) the successful transfer of HTML documents and their associated files. Users of the WWW (as with any network) must be prepared to accept these occasional delays or downtimes.

- **Server host capabilities.** The computer system, or systems, that host both the web server and the database server will have certain capabilities based on the CPU, amount of memory, amount of hard drive space, network interface, and the specific operating system. Any organization considering supporting database access through the WWW, or an intranet, should plan to provide one or more computers, capable of supporting the needs of the web server, the database server, and any other software processes each machine may run.

- **Server workload.** With reasonable planning, the computer, or computers, hosting the web and database servers should be able to manage normal or

expected high traffic usage. However, an unexpectedly high rate of user (client) requests could degrade the host computer's performance, and, subsequently, the performance perceived by users of the database.

- **Type and volume of data requested from a web site.** While data in the form of text is generally transferred very quickly, graphics, audio, video, and applet files may take a considerable amount of time to transmit (depending on the communication channel), thereby consuming extra system resources, and degrading perceived performance.

2.3.2 Methods for Providing Database Access

The WWW is capable of supporting a variety of remote computing applications. The technology behind these applications continues to change as the capabilities of the WWW evolve. The discussion that follows covers several current methods and technologies used to create applications that support database access through the web—namely *hand-crafted* HTML pages, the Common Gateway Interface, and Java.

2.3.2.1 Hand-crafted HTML Pages

The first generation of so-called WWW applications simply involves collections of HTML documents connected by hyperlinks [Baker96]. These relatively simple applications provide one-way communication of their information to the user when displayed on the client's browser. Each document of these applications is a hand-crafted, static HTML file stored in a computer file directory of the web server host computer. An enhancement to these first generation applications is the ability to embed multimedia resources—such as graphic images, audio, and video—as well as applet programs within the static HTML pages. As a result of the limitations of these manually-built, static HTML pages, the only way they can provide (pseudo) database access is by the creator of each page specifically incorporating database information within the static HTML.

2.3.2.2 Common Gateway Interface (CGI)

Second generation WWW applications overcome some of the limitations of hand-crafted WWW pages by offering the ability to dynamically create HTML pages in response to user

18

requests. This dynamic creation is accomplished through the use of the Common Gateway Interface (CGI) [NCSA-CGI]. CGI is a standard for interfacing external applications with information servers, or more specifically, web servers for this discussion.

A CGI program is executed in real-time, allowing it to dynamically process information. CGI programs can be written in any programming language that executes on the host computer, provided the language has the features needed to generate the desired dynamic web pages.

The steps involved in web-CGI interaction begin with the remote database user either selecting a hyperlink or submitting a filled out form from the document currently being displayed by the web browser. (Refer to Figure 4 for a model depicting the relationship among components when using CGI to support WWW applications.) The client software (web browser) then transmits the user input, from either the selection or form, to the web server as embedded text in the client's URL request. The web server receives the URL text, identifies the URL as a request to execute a CGI program, and executes the external CGI program (usually located in the "cgi-bin" directory). The web server also passes the embedded CGI information, which the CGI program processes to generate the new HTML page.

The CGI program may immediately generate the new web page and return it to the web server. However, in many instances, the CGI program communicates with another program that produces the needed data, which for this discussion is the database server.

In the specific case of a CGI program providing database access [Whetzel96], the CGI program formats the appropriate SQL query, connects to the database server, and transmits the SQL query to the DBMS. The database server executes the query and outputs the results. The CGI program retrieves the results of the database query, formats it using the appropriate HTML code, and sends the new web page back to the web server.
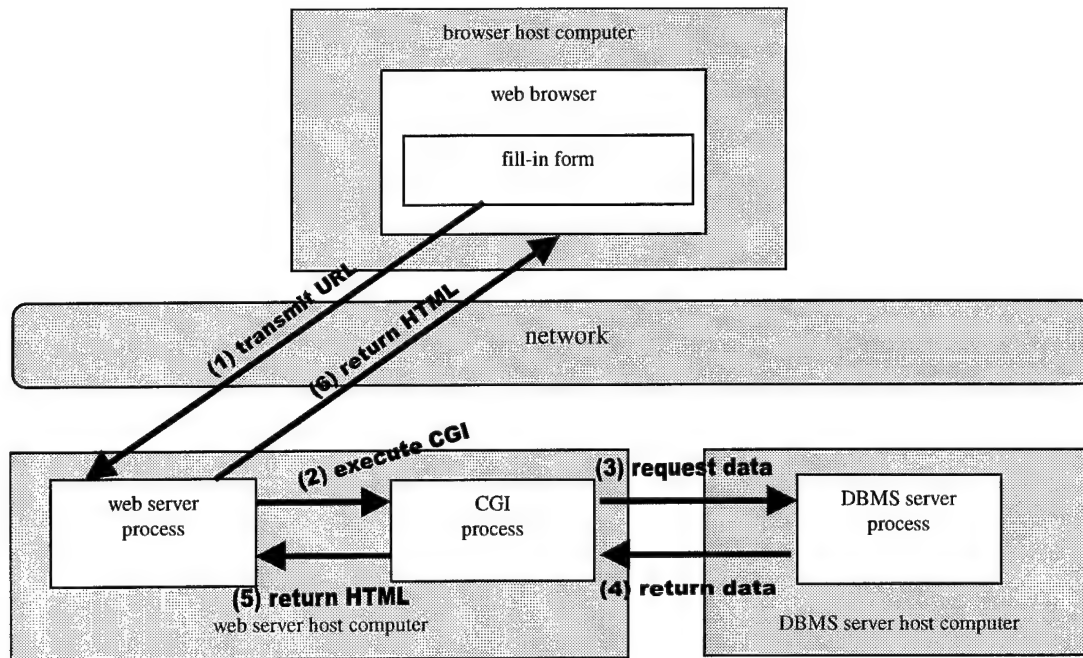
Figure 4. CGI in the WWW-DBMS Architecture.

- **Types of CGI Gateways**

There are four major versions of the CGI gateway [Whetzel96]. The first version, the Unix/SQL gateway, supports calls to a database on a Unix operating system platform. To support the Unix/SQL gateway, several RDBMS vendors provide programs and libraries that CGI programs can use to access the database.

An ODBC (Open DataBase Connectivity) [ODBC96] gateway operates similar to the Unix/SQL gateway with the CGI program accessing the DBMS through ODBC calls, rather than by using vendor-specific methods. ODBC is a Microsoft-developed open standard that provides a common SQL-based database interface to all ODBC-compliant DBMSs—of which there are many[9].

---

[9] Microsoft produces several ODBC compliant DBMSs, including FoxPro, Access, and MS SQL Server. Many other non-Microsoft DBMSs also support ODBC through software drivers provided by each manufacturer or by third-party software vendors. As a result, DBMSs that are accessible via ODBC—either through direct compliance or third-party drivers—are available on most major computing platforms, including MS-Windows NT, MS-Windows 95, Macintosh, and most Unix versions.

20

Perl gateways make use of the Perl interpreted scripting language. Perl is available for and operates on both Microsoft Windows and Unix operating systems. Several Perl code libraries exist that help create CGI programs. Also, extensions to Perl have been designed specifically for access to several DBMSs, including Oracle, Sybase, and Informix on Unix platforms. Perl programming is considered to be simpler and faster than C programming for similar web-to-database tasks.

Database-enhanced web servers have built-in database connectivity. In this version of a CGI gateway, the web server is largely DBMS-dependent. However, Microsoft-based web servers support ODBC compliant databases, allowing communication with a wider selection of DBMSs. Since the web server provides the database connection, CGI programs need only provide the specific application functionality. The CGI program then uses the specialized calls to the web server, which takes care of communicating with the database.

Some versions of database-enhanced web servers allow direct embedding of SQL calls within HTML documents. These products that support embedded SQL may still rely on external CGI programs—for example Sybase, Inc. offers the web.sql product [Liederman96]. However, DBMS vendors are beginning to release products that support embedded SQL without relying on CGI, but instead use their own mechanisms to process the embedded query language. For example, Microsoft has developed the Internet Server Application Programming Interface (ISAPI) [Gross97] as a direct alternative to using CGI with its line of server products [SQL Server96]. Also, Oracle's web server currently relies on CGI to execute its application programs that dynamically create Oracle database web pages, but that company plans on eliminating the CGI dependency in later web server releases [Montinola96].

- Drawbacks with CGI

While the Common Gateway Interface has made the development of a wide variety of web applications possible, it still has shortcomings. Some of these concerns apply to any applications using CGI while others are more specific to database access.

One problem with the CGI method of database access is that the web server must spawn (execute) a new process for each URL request requiring CGI execution [Montinola96]. For web sites that process high volumes of traffic, this will create a heavy burden on the host computer and slow down its performance.

Another problem with using CGI involves validation of the remote user input [Hamilton96a]. Any information for the database query that is sent from the client—such as text strings or numeric values entered into a web browser form—cannot be validated until the data reaches the web server. If the web server (or, subsequently, the CGI program or database server) finds invalid information, it must return an appropriate message to the client. The time delay for such validation, error feedback, and error correction will lead to an unresponsive user interface.

Also related to validation is the processing cost of having the CGI program perform data validation. In this case, if the form is complex and the number of database users is large, the server computer may be overburdened by the additional validation processing [Hamilton96a]. (Recall that the same host computer may be running the web server, CGI programs, database server, and possibly other processes.)

An additional drawback with the use of CGI is the potential bottleneck that may occur between the client and the database server. Some database access applications require a higher level of performance, and database vendors have worked to develop more effective client/server communications. The intermediate step of processing by the web server and CGI program will introduce communication delays. Also, the stateless nature of the WWW requires a separate login and logout procedure for each transaction, introducing additional delays. This overhead

22

imposed by placing the web server and CGI program between the client and the database server may result in unacceptable performance.

Another large area of concern with the use of CGI is security. While there is nothing inherently insecure about CGI, poor design of CGI programs or improper reuse of existing CGI scripts can lead to security problems [Stein96].

2.3.2.3 CORBA

The Common Object Request Broker Architecture (CORBA) is a standard developed by the Object Management Group (OMG) that promotes interoperability among the ever-increasing number of hardware and software products available today. The goal of CORBA is to allow applications to communicate with one another no matter who has designed them or where they are located—whether on a large network or on the same computer.

Central to CORBA is the concept of an object request broker (ORB), which is described by the OMG [OMG96] as:

> the middleware that establishes the client/server relationships between objects. Using an ORB, a client can transparently invoke a method on a server object, which can be on the same machine or across a network. The ORB intercepts the call and is responsible for finding an object that can implement the request, pass it the parameters, invoke its method, and return the results. The client does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of an object's interface. In so doing, the ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems.

As depicted in Figure 5 (adapted from [Orfali97]), a client object is able to use an ORB to remotely invoke the methods (also known as *services*) of any server object registered with the ORB. Under CORBA, the interface for each distributed object is declared using the CORBA interface definition language (IDL). Any client requesting services of a distributed object has access to the services of that object only through the features (methods, attributes, exceptions, etc.) defined by its IDL interface.

Figure 6 (adapted from [Orfali97]) depicts the components of an ORB under the CORBA 2.0 specification along with their relationships to a client and a server object implementation. Orfali and Harkey [Orfali97] describe the name and basic role of these components as:

- Client - that which requests the services of a distributed object.

- Object Implementation - the implementation of the distributed object. Provides the services (functionality) of the distributed object.

- Interface Repository - a database of metadata describing the available distributed objects and their interfaces.
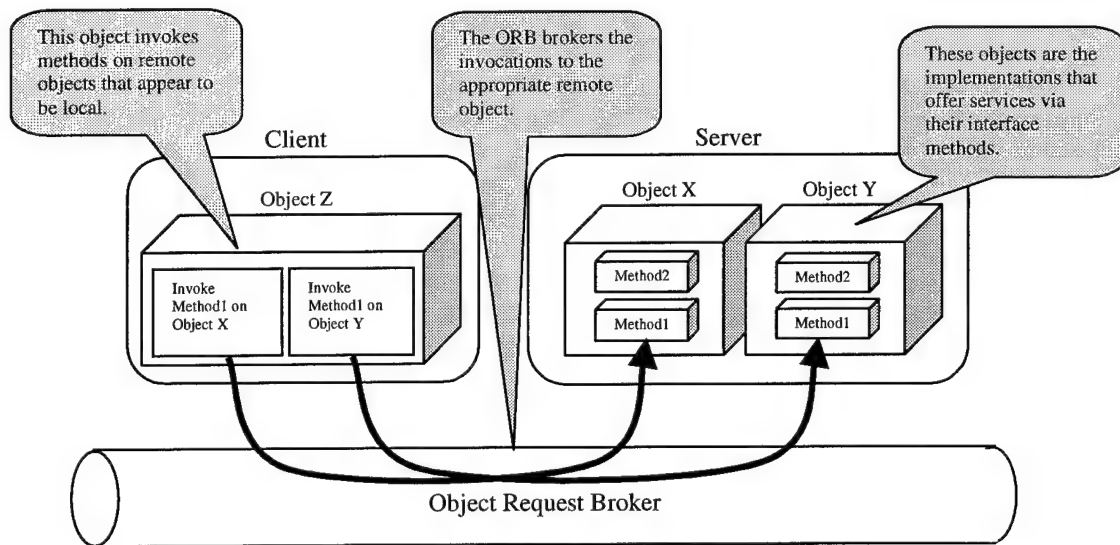
Figure 5. Invoking a method on a distributed object via an ORB.
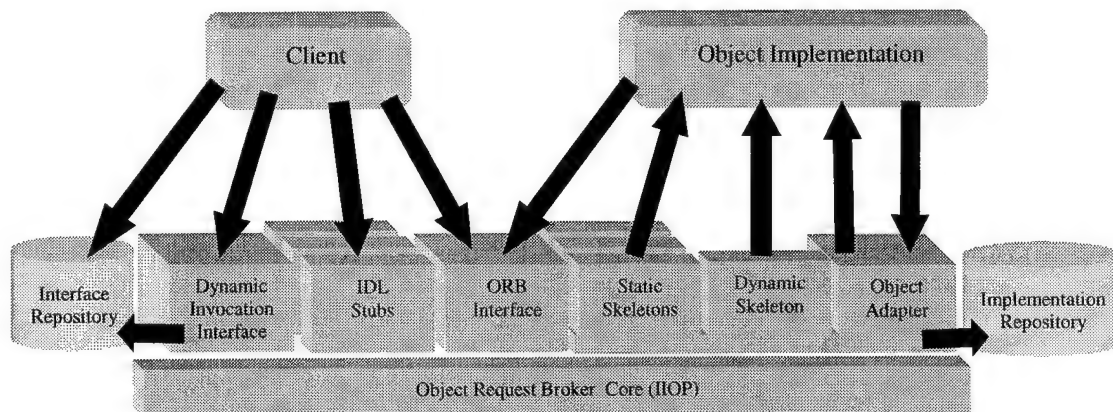
Figure 6. Components of an ORB.

24

- Dynamic Invocation Interface - interface to allow access to objects and services that are discovered by the client at run-time (as opposed to static definitions of objects available at compile-time through client IDL stubs [see below]).

- Client IDL stubs - collection of pre-compiled stubs representing the static IDL interface definitions of distributed objects available to any client. The stubs allow a client to reference the services (methods) of a distributed object at compile-time.

- ORB Interface - APIs available to clients and object implementations that allow access to the services of an ORB.

- Static Skeletons - similar in their role to stubs on the client side, these represent the compiled interfaces that the ORB uses to access the object implementations in response to client requests.

- Dynamic Skeleton Interface - allows an ORB to access objects at run-time that do not have compiled static skeletons.

- Object Adapter - run-time environment that provides communication between the server objects and the ORB core. Performs such tasks as passing requests, instantiating objects, and assigning object identifiers.

- ORB Core - central component of the ORB, providing the services to communicate with the other ORB components as well as other ORBs. Internet Inter-ORB Protocol (IIOP) is the OMG defined protocol to allow support for ORB services over TCP/IP[10] networks.

Also important to CORBA are the *CORBAservices*, which define features to support distributed objects that augment and complement the functionality provided by the ORB. Standards for sixteen object services have been published by the OMG. Two such CORBAservices are the *Naming Service*, providing support for objects to locate each other by name, and the *Security Service*, providing a framework for the security aspects of using distributed objects.

To reap the benefits of CORBA, distributed server objects will need to be available that conform to CORBA, client objects must be designed to use the available distributed services, and ORBs must be in place to broker the activities. At such time, client applications, such as WWW

---

[10] TCP/IP is the network protocol suite used on the Internet. TCP (Transmission Control Protocol) is a transport layer protocol, providing connection-oriented and stream-oriented services over IP. IP (Internet Protocol) is a network layer protocol, supporting connection-less, best-effort packet-switching through a network [Malkin96].

browsers and remote applets, will be able to directly request the services of remote objects through the Internet. Consequently, the resources of CORBA-compliant DBMSs will be accessible to these remote clients, as well as the services of other CORBA-compliant distributed objects scattered throughout the nodes of the Internet. However, CORBA, like the WWW, has only been available for a few years, so both research into and application of CORBA is still limited.

As more systems—including DBMSs—become CORBA-compliant, having a standard method to access distributed objects across the Internet should be a boon for distributed computing. The potential benefits that CORBA may bring to database access through the WWW (or through any network) go beyond what can be described in this literature review. (Note that CORBA will be discussed, again, later in this chapter during the presentation of Java's features.)

Microsoft also provides an environment for distributed objects, called the Distributed Component Object Model (DCOM). While DCOM offers many of the same features as CORBA—e.g., there is a Microsoft IDL and a mechanism comparable to an ORB, a fundamental difference is that DCOM is designed only to work on platforms using Microsoft operating systems. Orfali and Harkey provide a description of DCOM along with a side-by-side comparison of its features to those of CORBA [Orfali97: 287-337]

- CorbaWeb

Some research has been focused directly on integrating CORBA with the WWW. Merle, Gransart, and Geib [Merle96] suggest that the combination of the WWW and CORBA will become the next infrastructure for distributed client/server computing. In their study, they developed and implemented an environment, called CorbaWeb, that allows access to CORBA objects through the WWW. Their CorbaWeb environment, however, still relied on the use of CGI, so any issues associated with CGI still apply.

2.3.2.4 Java

Another approach for providing web access to databases is an application of Java and the rapidly expanding feature set it provides. Java, or more specifically, *The Java Programming Language Environment* [Gosling95], provides not only a programming language, but a rich (and growing) set of libraries to support complex development needs and a run-time environment that directly addresses issues of software portability and deployment. A more in-depth discussion of the features of Java and the potential it has for supporting web-based access of databases follows.

2.3.3 Java's Role in Database Access through the WWW

Java provides many capabilities in support of a wide range of computing needs. In order to better comprehend the capabilities of this new programming environment, the next few paragraphs present an overview of Java's general features and the support it provides for multi-platform operation, applet programming, and networking.

2.3.3.1 General Features

Many people assume that the name Java refers to "just another programming language," even though it encompasses a much wider range of features. While this is a narrow perspective, the programming language aspects of Java certainly warrant closer inspection as part of the overall discussion of Java's capabilities.

Java combines many of the characteristics of previously existing languages [Gosling95]. First, Java's syntax is very similar to that of C++, which eases the "learning curve" for programmers already familiar with C++.

Second, Java is an object-oriented programming language, drawing on many of the concepts introduced by the languages Eiffel, SmallTalk, Objective C, C++, and, most recently, Ada95. Third, the Java language does not support pointers, thereby, eliminating many of the programming difficulties (bugs) that appear in software as a result of poor pointer management. Instead of using pointers, references to objects are made using symbolic "handles".

27

Fourth, Java manages its own memory deallocation through *automatic garbage collection*—a method that lets the Java environment, rather than the programmer, identify unreferenced objects and return the memory to the available pool. Garbage collection is a concept familiar to many Lisp programmers, but it is utilized in a more efficient manner in Java.

Meanwhile, Java specifies the sizes of all of its primitive data types—for example, a *short* integer is always represented by 16-bits. This explicit type specification directly supports Java's other multi-platform capabilities (discussed below), by guaranteeing that each primitive data type will be implemented the same on every hardware platform upon which Java programs execute. Also, Java does away with ASCII 8-bit character form and, instead, uses the 16-bit Unicode character representation. Using the Unicode representation supports easier implementation of the same software product for different written languages.

Despite the recent introduction of Java, a comprehensive set of Application Programming Interfaces (API) have already been defined for this programming language. Standard class libraries cover the APIs for basic language objects and fundamental types, file and stream I/O (input/output), container and utility classes, and platform-independent graphical display objects (supporting X Windows, Microsoft Windows, and Macintosh user interfaces). Additional APIs have been defined that support 2D and 3D graphics, telephony, 2D animation, multimedia (audio, video, and MIDI data), shared whiteboard and collaborative applications, network and systems management, electronic commerce, and encryption and authentication. (JDBC, the API for relational database access, will be discussed later in this section.)

Finally, each Java program may be compiled either into (1) a platform-specific, stand-alone executable form or (2) a platform-independent form that may be executed on a variety of computing platforms. The platform-independent version consists of *bytecodes* that may be executed by a Java virtual machine. Bytecodes and the Java virtual machine are discussed in more detail later in this next section.

- Java vs. JavaScript

Before ending the discussion of Java's general features, the distinction should be raised between Java and a related programming language called JavaScript. JavaScript was developed by Netscape Communications Corporation as a scripting language to provide enhancements to web pages displayed in web browsers. JavaScript programs are embedded directly in HTML pages as a collection of program statements. Figure 7 provides an example of JavaScript script as it would appear in an HTML page. Figure 8 shows the resulting image displayed by the web browser. Such scripts are executed either by the client web browser or by the web server, depending on whether the scripts are client-side or server-side versions [Javascript96].

```
</SCRIPT> </HEAD>
<BODY>
<FORM>
Enter an expression:
<INPUT TYPE="text" NAME="expr" SIZE=15 >
<INPUT TYPE="button" VALUE="Calculate" onClick="compute(this.form)">
<BR>
Result:
<INPUT TYPE="text" NAME="result" SIZE=15 >
</FORM>
</BODY>
```

Figure 7. Example of JavaScript in an HTML page.
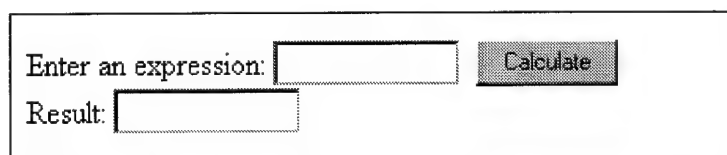
Figure 8. JavaScript form as seen in web browser.

The JavaScript language has similar keywords, syntax, and grammar to the Java language. However, JavaScript is object-based rather than object-oriented, which at a minimum means that JavaScript does not support inheritance. Also, JavaScript does not allow the declaration of new object types (i.e., the programmer is constrained to using the types provided),

does not provide static typing or strong type checking, and is not compiled—all features distinctly different from Java.

2.3.3.2 Multi-platform Support

Java incorporates a robust set of features that provide cross-platform support [Gosling95]. By far the most heralded feature of Java is its ability to execute on multiple platforms. This ability is due to two facets of Java: (1) the concept of a "virtual machine", consisting of the Java interpreter and a Java run-time system that execute as a software layer on top of an existing operating system and (2) *bytecodes*: machine-independent code designed to execute on the Java virtual machine.

If Java programs are compiled into bytecodes, they must then be executed on a Java virtual machine[11] (JVM). The virtual machine acts as an intermediate layer that translates between the Java program bytecodes and the operating system calls of the host computer, as depicted in Figure 9 [Fritzinger96]. This virtual machine can, in fact, be an operating system running on the hardware [JavaOS96], a program running on top of the operating system, or a component embedded within another application that is, itself, executing on the system. This last method is the most widely recognized since the latest versions of popular web browsers include JVM components.

With the support of the Java interpreter and the Java run-time system within the Java virtual machine, these newer web browsers can execute small Java applications (applets), taking full advantage of the other features Java programs have to offer. And since a Java applet is executed by the Java virtual machine, the same Java applet will work within any and all web browsers that provide Java support.

---

[11] Rather than have the virtual machine translate the Java calls to the native operating system, some virtual machine implementations include a "just-in-time" (JIT) compiler [Morrison97: 976-984]. The JIT compiler compiles the bytecode instructions into native code that can be executed directly by the operating system. This compilation step improves execution speed of Java programs by varying degrees—on the order of from two to fifteen—depending on the specific program features.
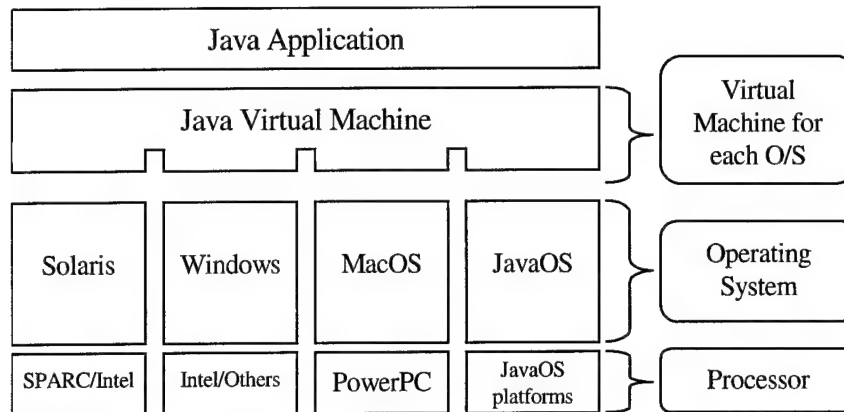
Figure 9. Model depicting the placement of the Java Virtual Machine.

2.3.3.3 Applet[12] vs. Application

A major advantage with Java applets is that due to their relatively small size—which can vary greatly depending on the design and features of the program—they can be used to enhance HTML documents accessed through the WWW. In the typical applet usage scenario, a reference that identifies the applet is placed within an HTML document—similar to the embedded references for graphic images, audio files, etc. The bolded text of Figure 10 provides an example of the same applet being embedded twice in a document [Criswick96]. Each applet section begins with the "<applet" tag and is terminated with the "</applet>" tag. The second instance shows the use of parameters that will be passed to the applet at run-time.

After the client web browser loads the HTML page, it requests the applet from the web server. The web server transmits the applet to the web browser, which uses its Java virtual machine component to execute the applet. If the applet has displayable features, they are shown within a specified area of the currently displayed web page. Figure 11 shows the resulting web browser image from the two running digital clock applets embedded in the HTML of Figure 10

---

[12] An applet, in the context of Java programming, is a small Java application that, typically, has limited functionality and is executed by the Java virtual machine component of a web browser [Hopson96].

[Criswick96]. Many such Java-based applets have been created that allow web documents to display animation, play sound, and interact with the user in real time[13].

```
<h3>Introduction</h3>

The following is a simple digital "LED" clock. Although you cannot change its size, you can configure the colours of the
digits, the frame and the background. I have made a few modifications since this clock was published in December '95 in
the hopes of making it a little more useful and easier to configure and setup.<p>

Questions or comments to: <a href="author.html">The Author</a> by e-mail to <a
href="mailto:author@website.net">author@website.net</a>.<p>

<hr>
<center>
<applet codebase="classes" code="BigTime" width=120 height=36 align=absmiddle>
</applet>
<p>
<applet codebase="classes" code="BigTime" width=120 height=36 align=absmiddle>
<param name=ledcolor value=black>
<param name=framecolor value=lightgray>
<param name=backcolor value=lightgray>
</applet>
<p>
</center>
<a name=source>
<hr>
<h3>Source code for the clock</h3>
Only one file makes up the source code for the clock:<p>

<pre>   <a href="BigTime.java">BigTime.java</a>
</pre>
```

Figure 10. HTML text behind web page in Figure 11.

- **Applet Security**

A concern that developed with the use of Java applets in web browsers was the possibility that an applet downloaded with a web document could affect the client computer in undesirable ways, either by design or by accident. For example, if so designed, the downloaded applet could read information from, modify, or even delete local files. After all, each applet is an independent program that executes on the client computer and could, potentially, access any of that computer's resources.

Another security concern was that a Java applet could be produced by a "hostile compiler"—one that doesn't subscribe to the "safety" features incorporated into the Java

---

[13] Several Internet web sites provide links to freely available applets. JavaSoft, the subsidiary of Sun Microsystems responsible for developing and maintaining Java, provides an index to several of these Java applet directories at "http://www.javasoft.com/nav/applets/index.html".

specification [Gosling95]. Therefore, additional security measures were implemented in the Java specification to restrict the access of downloaded applets from the rest of the host computer [Applet Security96] and to ensure that only applets complying with the Java specification are allowed to execute [Gosling95]. Such restrictions include limited or no access to local or network files, print devices, or Java class files. Upcoming version of web browsers, however, are expected to support multi-level security for applets (i.e., finer grained control over which resources an applet may access).
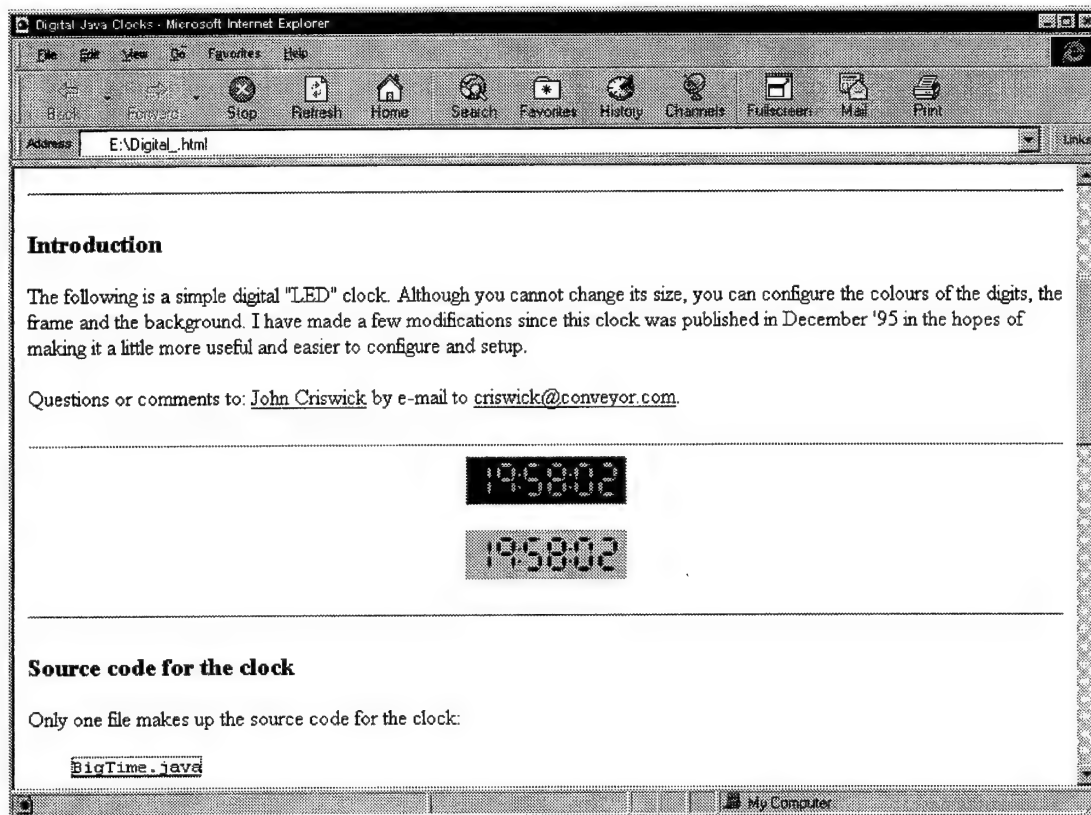
Figure 11. Resulting web browser image from HTML-based applet.

- Software Deployment and Applets

Related to the use of applets is their potential to improve software deployment. By creating new and revised software products as applets, software developers can use a web server

33

for deployment. By using a web server for deployment, remote users will automatically receive the latest version of each applet whenever they access the web page containing its reference.

2.3.3.4 Built-in Networking Features

Java was designed from the very start to support development of applications that will run in a heterogeneous, network-wide distributed environment [Gosling96]. The Java core API incorporates direct support for creating and manipulating URLs, as well as the ability to address network resources via their URLs (if available and assuming security restrictions permit). Also, Java supports point-to-point communication between client and server applications using sockets[14]. With socket connections, session-oriented communication can occur between client and server software processes.

With these two features alone (URL and socket support), a Java application that has access to the WWW or an intranet can make use of numerous network resources. When combined with the other features already described that provide for security, multi-platform support, and applet development and deployment, the Java environment provides software developers with an extensive set of tools for Internet and WWW application development.

2.3.3.5 Servlets

Recognizing the popularity of the CGI approach to providing support for web sites, the developers of Java created an alternative to CGI, called *servlets*. Servlets are Java programs that are designed to fulfill much the same role as CGI but utilizing the features and environment of Java. Typically, web servers invoke servlets to accomplish external processing and dynamically create new web pages—e.g., a servlet can request information from a database and then generate and return a web page containing the data formatted in a useful manner. From the web browser perspective, servlets behave exactly like CGI programs in that they return dynamically generated

---

[14] A socket is one end-point of a two-way communication link between two programs running on a network [Java Tutorial97].

HTML pages. Consequently, servlets, as with CGI, perpetuate the stateless nature of WWW interaction.

2.3.3.6 JDBC for RDBMS Access

While the list of features discussed above for the Java language environment will support many application development needs, an additional specification and set of Java classes was seen as necessary to more directly support access to RDBMSs under the client/server paradigm. Since many of the existing databases that users wish to access through the web are relational, direct support for communication between Java programs and RDBMSs has been a high priority with many businesses. As a result, the JDBC (not an acronym) specification was developed to be a low-level API that allows relational database access from within Java applications and applets [Hamilton96b]. JDBC was developed as a *call-level* SQL (Structured Query Language) interface for Java, which means a programmer can use the JDBC API to embed SQL—the standard relational database access language—statements directly within Java code. JDBC's design has the same roots as Microsoft's ODBC (refer to Section 2.3.2.2).

The primary advantage provided by JDBC is the ability to communicate directly with a DBMS from within Java programs—in and of itself, not a unique capability since many other programming languages already offer database extensions. However, when combined with the networking features of Java, JDBC allows developers to build applications that connect directly to remote RDBMSs through a network using the available features of SQL. By also factoring in the ability to create applets using Java, a developer can create remote database access programs that are deployable to any WWW or intranet user.

Figure 12 depicts the role of JDBC in applications employing database access through the WWW. The web server supplies the applet to the web browser, and the applet uses the JDBC API to establish a connection to and communicate directly with the DBMS.

35
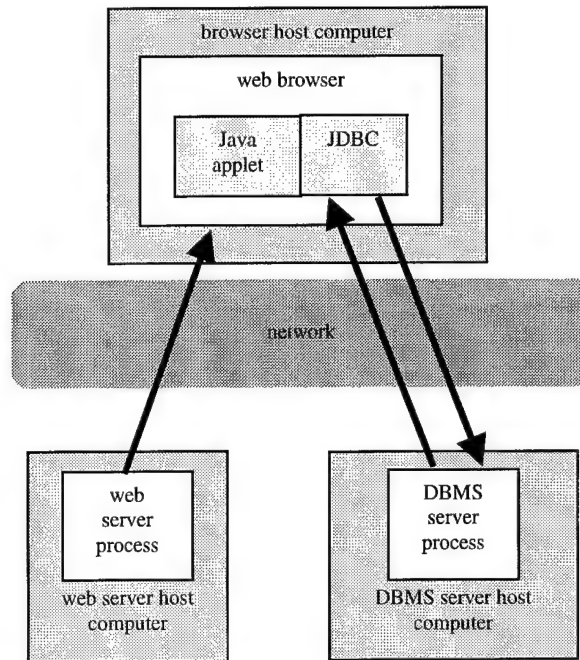
Figure 12. Role of JDBC in Supporting Web Access to a DBMS.

Other performance advantages [Hamilton96a] appear when considering the use of Java and JDBC compared to CGI (recall the CGI shortcomings discussed in Section 2.3.2.2). First of all, Java client applets can perform the validation of user input, thus eliminating the validation burdens on the server side and the subsequent transmission delays for feedback if invalid data appears. Also, the ability of a remote Java applet or application to connect directly with a DBMS eliminates the potential web server bottleneck between the client and database server.

2.3.3.7 Java and CORBA

Combining the features of Java with those of CORBA expands the capabilities of each. First, Java provides highly portable code, thus allowing CORBA-compliant server and client objects to be highly portable. Second, CORBA allows Java clients to transparently access the

36

services of remote objects created with any programming language—not just those created with Java[15].

As a result of the integration of Java and CORBA, distributed software systems can be designed with "thin" Java clients that use the services of objects executing on more powerful hosts. Meanwhile, CORBA server objects, if created using Java, can be executed on numerous platforms, supporting greater flexibility in the choice of operating environments for the distributed system.

Under a scenario for remote database access using CORBA, a remote Java application (or HTML document-embedded Java applet) would use the ORB to locate the desired CORBA-compliant DBMS whose services are desired. The Java application would then establish communications directly with the DBMS and request its services.

- **Java ORB Availability**

Several vendors have released CORBA 2.0 standard compliant ORBs that support Java applications and distributed objects. These products include development features that allow programmers to build applications that use the services of CORBA ORBs and CORBA-compliant distributed objects. Three such ORB products are *Joe* [Joe97] (not identified as an acronym) by Sun Microsystems, *OrbixWeb* [Iona97] by Iona, and *Visibroker for Java* [Visigenic97] by Visigenic.

Other research has also been performed into providing CORBA support for Java. Duan [Duan96] developed a software tool called HORB, based on the concept of the Object Request Broker, for creating Java client and server objects. Through his research, Duan was able to demonstrate a methodology for developing distributed database applications on the Web using

---

[15] With the release of the JDK 1.1, Sun provided the Remote Method Invocation (RMI) API that offers many of the features available through CORBA. However, the most notable difference between RMI and CORBA is that RMI only supports communication between client objects and server objects executing inside Java virtual machines--i.e., RMI only allows Java objects to use the services of distributed Java objects.

Java. As a result of his work, the author concluded that using a Java approach for distributed database application is more flexible, portable, scaleable, and robust than the CGI approach.

2.4 Existing Web-based Distance Education Systems

Several web-based distance education systems are available commercially or are being researched at educational institutions. This section will provide a brief summary of the research being performed and the course management features these systems provide.

2.4.1 University Research

The Virginia Polytechnic Institute and State University has been investigating the use of a web-based testing system called QUIZIT [Tinoco97]. QUIZIT includes features to simplify quiz authoring, administer and grade quizzes, and store quiz results. This system relies on a Unix-based relational DBMS and C language CGI scripts to support database interaction. Student registration into the system was performed manually during the computer account setup.

Miranda and Pinto discuss use of WWW, e-mail, and newsgroups to support university courses [Miranda96]. This group's research primarily focused on posting course material on the WWW. In the future, they plan to perform on-line self-assessment, but they are currently only considering use of HTML with CGI.

Nishida, Saitoh, Tsujino, and Tokura describe their use of the WWW and e-mail to support existing university courses [Nishida96]. Lecture material and course updates are posted through WWW, and students ask questions and submit reports through e-mail. Their system did not support on-line registration or testing and did not employ Java technology.

The University of Illinois has been exploring the use of the World Wide Web, Java, and virtual environments in education [Singh96]. The school has established a dedicated web server that supports tutorial modules and other research information. The system's use of Java was limited to animation and some user interaction. Student access was maintained through flat file storage (no DBMS) and Perl CGI scripts were used to present much of the content.

Lockledge, Geister, and Mitchell discuss an architecture for web-based learning environment course management systems [Lockledge96]. The authors discuss how the primary users of a web-based learning system are students, instructors, and administrators. The authors comment on the importance of identity validation and suggest a two-pronged approach: (1) require access by user account and password and (2) only allow access by users working on computers with authorized IP addresses. Meanwhile, the authors list the following types of databases as those a course management system would require:

- Student Profile and Record
- Instructor Profile and Assignments
- Curriculum Definition
- Educational Materials
- Page Templates
- Schedules and Resource Assignments

Considerable research has been done at the University of Illinois at Urbana-Champaign in developing a web-based distance education system, called Mallard™, for presenting college courses [Swafford96a] [Swafford96b]. The Mallard environment provides a high level of security; course material, assignment, and test preparation utilities; on-line testing; automated assignment and test grading; and grade recording.

Mallard makes extensive use of CGI scripts to support its many features. Security is provided through the use of a secure HTTP protocol supported by both the web server and popular web browsers. Also, the password database is encrypted to prevent access to sensitive grade information.

The Mallard developers recognized that improper design of the system could lead to heavy workload for the HTTP server, primarily due to frequent server requests and CGI program executions. Therefore, they attempted to minimize the number of student requests to the server. They also suggested that the use of Java client-side programs would lighten the workload of the server and, therefore, intend to explore the use of Java applets in future work.

Another research group at the University of Illinois at Urbana-Champaign has developed the CyberProf asynchronous web-based distance learning environment [Hubler97]. The CyberProf system presents lecture notes on-line and provides discussion groups for communication between students and instructors. The system also supports homework and quiz presentation with solutions being submitted on-line for immediate and automated grading and score recording. CyberProf was implemented with Perl scripts and C routines and is undergoing continued development.

2.4.2 Commercial Products

Along with university research into developing these systems, several software vendors have released, or are planning to release, web-based distance education course management products. These vendors' products support a wide array of features covering both synchronous and asynchronous learning. Among these vendors is ILINC with its LearnLinc™ I-Net product [LearnLinc97]; WebCT, which began as a research project at the University of British Columbia [WebCT97]; Centra Symposium, with its all-Java implementation [Centra97]; TopClass™, based on the WEST project from the University College Dublin [TopClass97]; and LearningSpace by Lotus Corporation [LearningSpace97].

III. Analysis and Design Recommendations

This chapter provides an analysis of the operating environment of the ACSC distance learning system and presents a recommendation for improving that operation. First, a description of ACSC's web-based system operating environment is provided. Next, that operating environment is explored in the context of a proposed *Content Delivery Architecture*. Possible alternatives to improving ACSC's student data access shortcomings are then presented. And, finally, a recommendation to address student data access is provided, along with two software design alternatives that will support the recommendation. The complete design and implementation of the two software design alternatives is, subsequently, discussed in Chapter 4.

3.1 ACSC Web-based Distance Learning Environment

Section 1.3 discusses two of the immediate needs of the ACSC web-based course, and Appendix A outlines the long-term goals. This section describes the current web-based distance learning environment of ACSC (prior to offering complete on-line instruction) as the foundation for solution analysis in the sections that follow.

3.1.1 Configuration

The current configuration of ACSC's web operations (depicted in Figure 13) consists of the following hardware and software:

- an SGI Indy computer system operating as the web server host and maintained within the ACSC building,

- the IRIX operating system (a version of Unix) as the host computer operating system,

- Netscape Communications Corporation Enterprise version 2.0 web server software, and

- Perl language scripts supporting CGI to provide interactive content for students.

The web site is operated behind the Maxwell AFB base network firewall, and the web address (URL) is not advertised and not linked to any other Maxwell Air Force Base web page.

41

Access to the ACSC course web site is restricted through the use of account and password authentication on the web server. If the account/password combination is recognized and authorized, the web server transmits the appropriate web page.
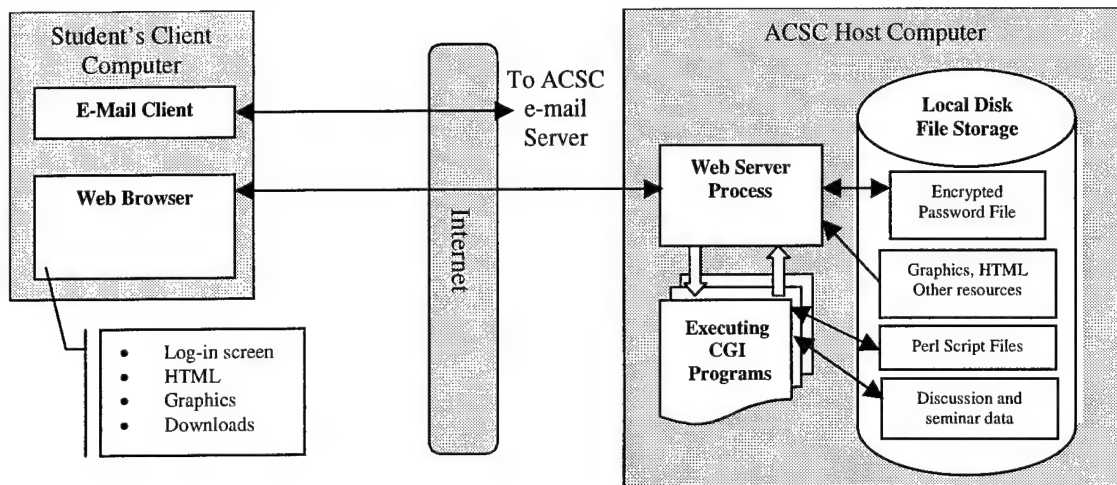
Figure 13. Current configuration of ACSC web operation.

3.1.2 Web Site Content

Several types of content are available to ACSC correspondence students through the web site. HTML files present text, images, and audio for the nine blocks of instruction. Meanwhile, text-based chat and bulletin boards systems support interaction with the web site and other students.

3.1.3 Course Registration Process

Registration of students into the ACSC correspondence course is accomplished through the following steps:

1. Student signs up for the course at the nearest Base Education Office.

2. Base education personnel enroll the student in the course via computer terminals that update the CDSAR (see below) Oracle database, which is operated by the Extension Course Institute.

3. ACSC Distance Learning Office personnel query the CDSAR database for new student and student update information through a Telnet connection and transfer the results back to their office using FTP.

4. ACSC Distance Learning Office personnel create web accounts and passwords for the students and manually enter them into the web server account/password database through the web server interface or via a utility feature provided by the web server.

ACSC staff perform steps 3 and 4 above twice weekly to update the student information for the ACSC web site [Vice97]. Querying the database and transferring the data (step 3) requires approximately ten minutes per session while manipulating the data (step 4) takes approximately twelve minutes.

3.1.4 CDSAR Features

CDSAR is an acronym for *Course Development Student Administration and Registrar*, which is a relational database of Air Force correspondence student information maintained by the Extension Course Institute (ECI). CDSAR contains the personal information, testing results, and status of students in the ACSC non-resident program. This database also tracks students in many other AF correspondence programs. Any change to information in the database, such as revising an incorrect test score, must be submitted via a request to CDSAR personnel.

CDSAR originally began as a file-based database. It was later converted to a relational database using Oracle version 5.2. While the RDBMS has since been upgraded (currently using Oracle version 7.2.2.3), the originally structure of the database has largely remained unchanged [Locklin97a]. As such, the database does not take advantage of many of the new features of RDBMSs that have been standardized over the last decade. For example, modern relational database design practice generally require the use of primary keys (to guarantee uniqueness) and foreign keys (to ensure referential integrity of relationships). However, these features have not been incorporated into CDSAR [Locklin97b].

Since integrity constraints are not designed into the schema, database integrity and consistency are enforced primarily by external application logic in the form of Oracle Forms and

43

C and COBOL programs as well as a large number of unique indexes on the database tables[16] [Locklin97a]. Therefore, batch processing is used to perform any insertions, updates, or deletions that affect multiple tables within the database. This aspect limits many remote users—those who cannot use the local database client interface—to having only read access since direct updates will by-pass the application logic enforcing database integrity and consistency.

3.2 Content Delivery Architecture

This section introduces a model, called the Content Delivery Architecture, that categorizes the features of a system that delivers content to a consumer. This architecture is subsequently used to provide a context for discussion of solutions to support ACSC requirements.

3.2.1 Overview

A review of the system employed by the ACSC staff for distance learning revealed several features that can be generalized to other systems that provide content of interest to end-users. Figure 14 shows the higher level components and the flow of data for this generalized *Content Delivery Architecture*. The five major components and their roles are as follows:

- **Content** – information of interest that is presented to the consumer.

- **Content Consumer** – end-user (human or other entity) that requires the content of interest.

- **Content Delivery** – system responsible for delivering content to the consumer, content viewers to the consumer, and requests to the provider.

- **Content Provider** – system that provides content and viewers to the consumer; responds to consumer requests; and manages all content, viewers and administrative data for the system.

- **Requests/Responses** – information from the consumer intended for the provider to assist in content delivery or maintenance.

---

[16] Note that relying on external application logic and unique indexes to ensure relational database integrity is not unusual and was the standard technique prior to the more recent releases of RDBMS products.
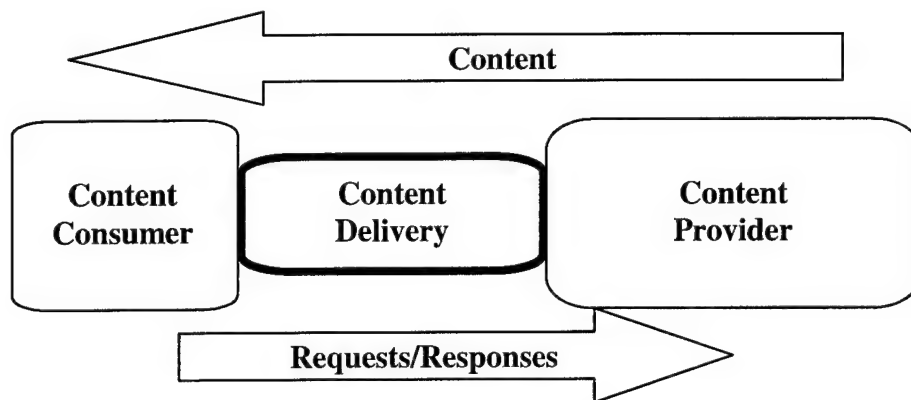
Figure 14. Components of the Content Delivery Architecture.

This architecture supports systems that are based on the electronic distribution of information as well the physical movement of information by other means. For example, the model may be used to represent delivery of the ACSC printed course material (or CD-ROMs containing the material) by the U.S. Postal Service as well as HTML documents via the Internet. The following sections describe each of the components of the content delivery architecture in more detail, with Figure 15 providing a graphical depiction of their location and relationships.

3.2.2 Content

Content is any form of data that is of interest to the consumer and is supplied by the provider. Content types are divided into several broad categories: (1) *display-only,* (2) *client-only interactive*, and (3) *client/server interactive.* Each successive category introduces another level of capability and complexity to the type of content, as described below.

- **Display-only Content**

Display-only content encompasses that information with which the consumer cannot interact or modify. In a traditional context, the text presented by a book to a reader provides display-only content since the reader (consumer) is only able to observe, and not interact with, the static text. Similarly, static text and images in web pages or non-hypertext electronic document viewers present display-only content. Additionally, the non-streaming audio and video that is

presented to a consumer via web browser helper applications (e.g., Java applets, ActiveX[17] programs, or other browser plug-ins) also constitute display-only content since the consumer cannot interact with or modify the content (i.e., audio or video) presented.
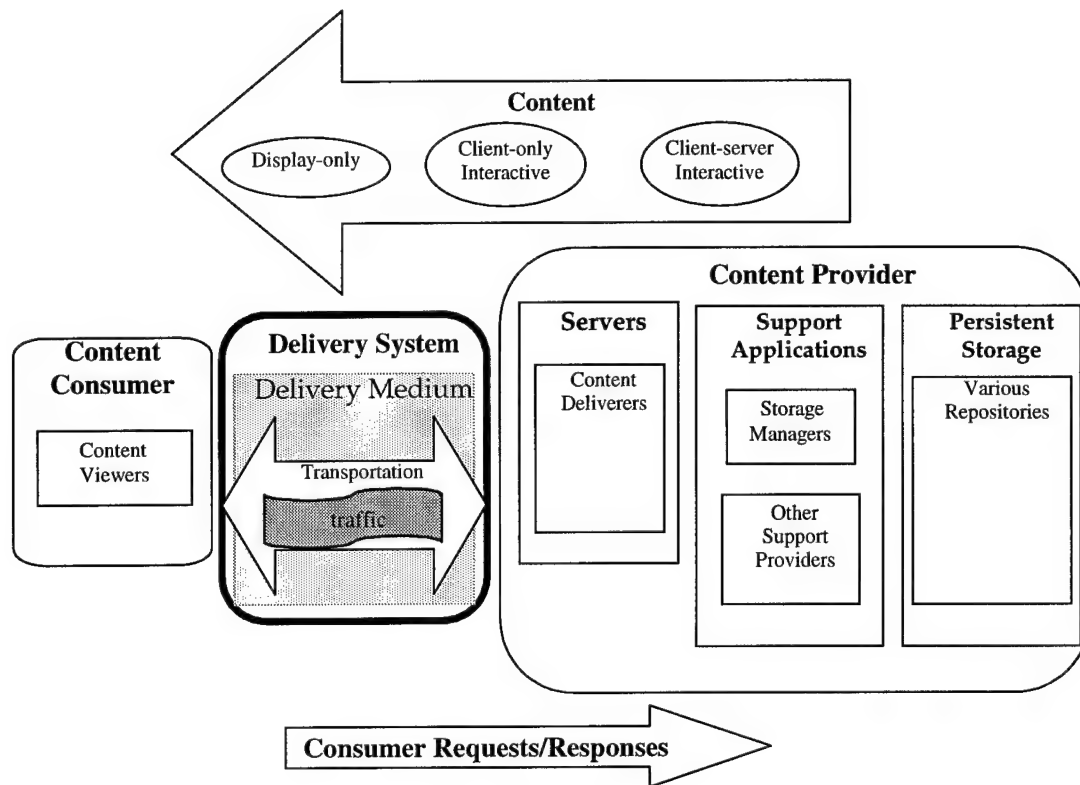
Figure 15. Lower-level view of Content Delivery Architecture components.

- **Client-only Interactive Content**

Client-only interactive content is that content with which the consumer can interact but that does not require resources outside the associated content viewer. In other words, the consumer can interact with (or affect the amount and type of) content presented, but no additional requests need be made to a content provider. In the context of the World Wide Web, *client-only* means that the web browser will not have to make additional requests to a web server. The

---

[17] ActiveX is Microsoft's name for a group of technologies that together provide many of the same features that have popularized Java. Chappell and Linthicum [Chappell97] provide a detailed discussion of the features of ActiveX.

content viewers that provide interactivity are web browser helper applications (applets, ActiveX, and other plug-ins) and the upcoming versions of web browsers that support dynamic HTML.

- Client/server Interactive Content

Client/server interactive content is that content with which the consumer can interact, and in doing so may send responses or additional requests to the content provider. In the context of the World Wide Web, HTML pages containing hypertext links allow the user to interact with the current content to choose which additional content should be presented. Also, web pages containing forms requiring user input are used to request new content from the provider. Applets, ActiveX applications, and other web browser plug-ins that communicate with remote servers constitute the types of viewers that support interactive content.

3.2.3 Content Consumer

The content consumer represents an entity that requests and views content from a content provider. The content consumer uses content viewers to interact with the content of interest. An example of a content consumer might be a student using a web browser to review course material available as HTML files. The primary category of sub-components within the content consumer system is *content viewers*.

- Content Viewers

A content viewer is a resource that presents content to the user. A content viewer may or may not support interactive content.

In the context of the World Wide Web, content viewers are typically web browsers. Web browsers directly support viewing HTML text and specific graphic formats (e.g., GIF and JPEG). They communicate across TCP/IP networks to remote web servers using HTTP. Web browsers also allow other viewers to execute within their environments. These other viewers are known as helper applications, and they assist with displaying those non-HTML objects whose references

47

are embedded within HTML pages. Helper applications typically are Java applets, ActiveX controls, and other plug-ins, which will be discussed next.

Applets are those Java applications designed to be executed from within a web browser—although they may also be executed through a utility program called *appletviewer*[18]. Applets are capable of presenting any form of content depending on the design of the particular program. Security restrictions imposed on applets by the host browser may, however, limit their functionality (see Section 2.3.3.3).

ActiveX controls are also programs that execute from within web browsers. ActiveX controls, like Java applets, are capable of presenting any form of content depending on the design of the control. The current security implementation for ActiveX controls allow the user to specify whether an ActiveX control is installed or not installed to run in the web browser. If the control is allowed to install, it has full access to the client computer's resources.

Other plug-ins are non-Java and non-ActiveX programs that also display content within the web browser. Such plug-ins are also capable of presenting any form of content depending on the plug-in's design. Security restrictions on these other types of plug-ins are limited to the user's discretion, as with ActiveX, on whether to allow the plug-in to load or not load.

Stand-alone applications in the context of this discussion represent programs other than those already mentioned that act as content viewers. Such applications are also capable of presenting any type of content depending on the design of the program. They may be capable of communicating with any resources with which they are designed to communicate. And security considerations depend entirely on the design of the program and the interface requirements of the provider's content deliverer (discussed below). Recall that Java programs may be developed as stand-alone applications or as applets.

---

[18] *Appletviewer* is provided with the Java Development Kit, a collection of Java classes, source code, and programming tools provided by Sun Microsystems.

3.2.4 Content Provider

The content provider is an entity that maintains content and provides that content to consumers. The content provider encompasses three categories of sub-components: (1) *content deliverers*, (2) *support applications*, and (3) *persistent storage*.

- **Content Deliverers**

  Content deliverers are components responsible for receiving and responding to consumer requests and delivering the appropriate content and, possibly, content viewers. A web server, for example, is a content deliverer that responds to HTTP requests and provides HTML files, along with embedded objects, to the requesting consumer. The embedded objects may themselves be content viewers for the customer—e.g., a document reader or a multimedia player. Other examples of content deliverers include streaming audio/video servers, e-mail servers, and JDBC network servers (discussed in Section 4.1).

- **Support Applications**

  Support applications are those software components that assist the content deliverer in creating and managing the content. The two broad categories of support applications are *storage managers* and *other support applications*.

  Storage managers store, deliver, and otherwise manage the persistent storage of content and other resources used by the content provider. Storage managers are, typically, either traditional disk file storage systems or database management systems.

  The other support applications that don't fit into the category of storage management perform additional processing tasks that fall outside the capability or tasking of the content deliverers. In the case of supporting web server content deliverers, these other support applications often take the form of one of the following:

  - CGI programs written with general purpose programming languages,

  - programs written using web server-specific APIs (e.g., ISAPI from Microsoft and NSAPI from Netscape), and

- Java servlets (see Section 2.3.3.5).

- **Persistent Storage**

Persistent storage acts as the repository for all non-transient data (including software) used by the content provider. Persistent storage is available in various forms that provide different levels of flexibility and performance for the types of data they store. For example, a file directory structure on disk can, generally, store any file type that contains binary data. Meanwhile, relational databases and object databases support more sophisticated storage, retrieval, and management of various data types. Some common forms of data and software to be stored include:

• HTML	• Applications	• Applets
• Audio	• Distributed objects	• Servlets
• Video	• Objects(non-distributed)	• CGI programs
• Graphics	• Relational Records	• Other raw data

3.2.5 Delivery System

The delivery system is responsible for transporting content from the provider to the consumer and requests or responses from the consumer to the provider. The delivery system is composed of the *delivery medium,* the *transportation system,* and the *traffic.*

- **Delivery Medium**

The delivery medium is the channel that supports transport of the content, viewers, requests, and responses. In the example of delivering printed course materials, the delivery medium might be the interstate highway system. In the context of the WWW, the delivery medium is the Internet, consisting of interconnected TCP/IP networks.

- **Transportation**

The transportation component of the delivery system is used to transport the content across the medium. For printed materials, the transportation might be the delivery vehicle of a

postal delivery system. For web pages, the transportation is the socket connections that stream data across the TCP/IP networks of the Internet and intranets.

- Traffic

    The traffic of the delivery systems is the content, viewers, requests, and responses already discussed. Content and/or viewers are transported from the content provider to the content consumer while requests and responses are transported from the consumer to the provider.

3.2.6 Requests/Responses

Requests and responses represent those messages sent from the consumer, via a content viewer, to the provider. The messages may either request additional content or supply the provider with information needed to maintain the content.

3.3 Distance Learning Student Management Under Web-based System

In the context of the discussion of Air Force correspondence student and course management systems, the following mapping is proposed under the Content Delivery Architecture. The students and distance learning staff (e.g., ACSC personnel) represent consumers; the course material, tests, other teaching or administrative resources represent content; and the combination of the computer systems operated by ACSC and ECI represents the content provider.

Under the current system being used for ACSC web-based distance learning, administrative data about the student consumers is stored as relational records in a relational database management system (ECI's CDSAR). Meanwhile, the course content is stored as HTML pages and associated multimedia files. Testing content is still managed as paper products mailed to the consumers, and test results are managed as relational records in the CDSAR database.

In the distance learning system just described, the role of CDSAR as the central repository for student data is expected to continue under any new Air Force model of distance

learning. Therefore, access to the relational data pertaining to enrolled students will continue to be a key design issue for any new system.

The goals of the ACSC staff to have a more full-featured web-based distance learning system require that the staff have more direct access to the student records currently stored in the CDSAR database. More direct access to the student data will support more capable student and course management systems for any Air Force web-based distance learning provider. Consequently, the focus of this thesis effort is directed toward examining new technologies, as well as ways to employ those technologies, to provide that access to the CDSAR data.

The following subsections describe various alternative approaches to providing access to the student records currently maintained in the CDSAR database. These approaches address the need from a larger, enterprise system perspective to emphasize the support for other existing (or potential) Air Force distance learning providers.

- **Enhanced CDSAR with Indirect Access**

In this approach the students are consumers of ACSC distance learning course content and student data content provided via ACSC. (See Figure 16.) Meanwhile, ACSC acts as both provider of course content and student data content to the student consumer and, is itself, a consumer of CDSAR student data content from ECI. In general terms, the distance learning content provider continues to look to ECI's CDSAR database for storage and maintenance of student data and requests student data content for both itself and its consumers (student customers). ECI as a student data content provider is, as a consequence, responsible for providing the interface features for consumer access—in this case only for ACSC. Figure 17 shows the relationship of the entities and software components involved with this approach.

52
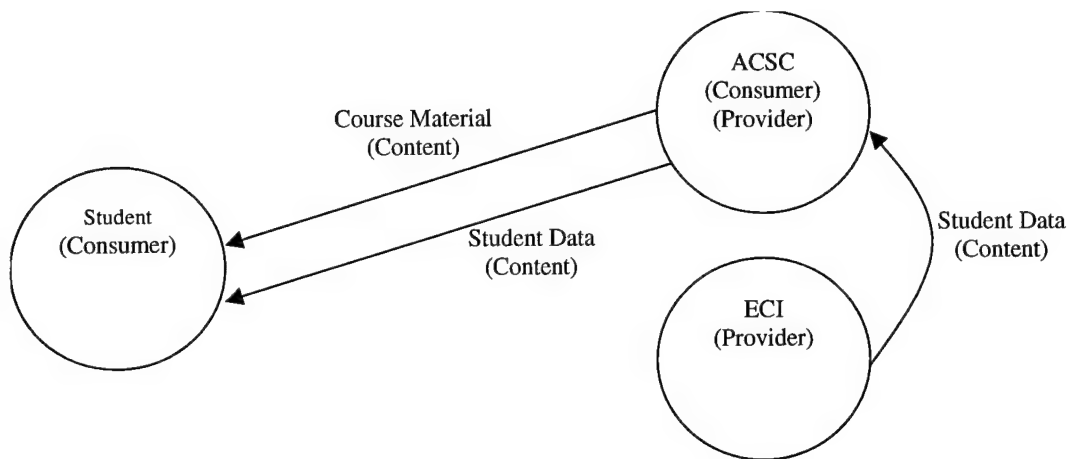
Figure 16. Consumers and providers for enhanced CDSAR with indirect access.
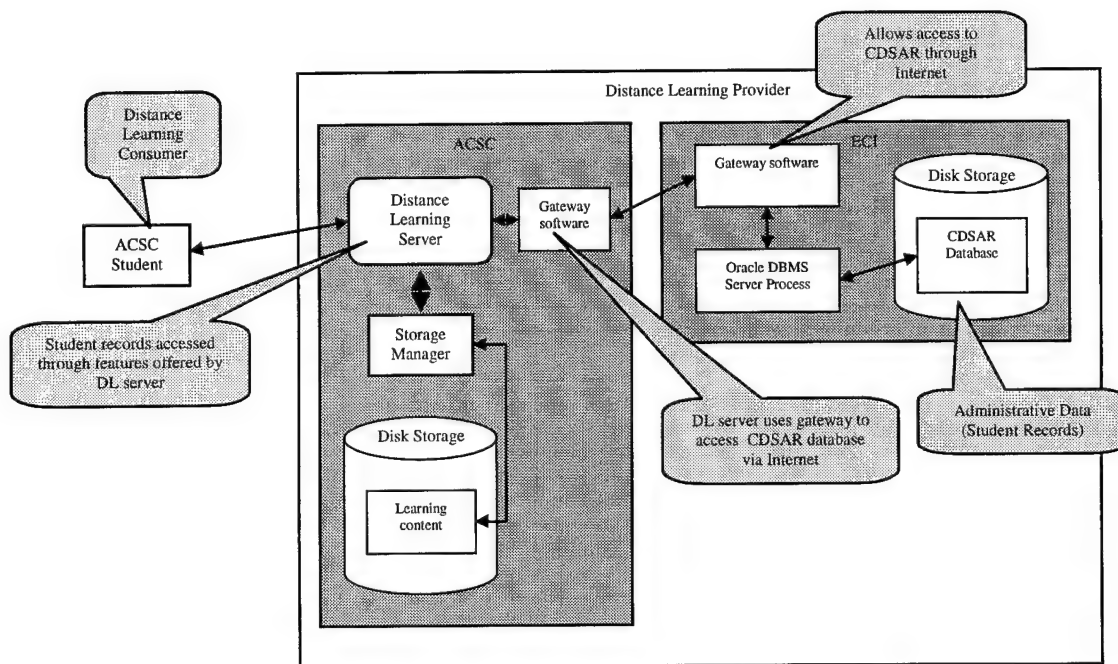
Figure 17. Relationship of the components comprising the indirect access approach.

The approach of indirect access to the CDSAR content offers the advantage of centralized control of the data. With centralized control, only one geographical location must maintain the resources—i.e., staffing, equipment, and software—necessary for database administration. Also, centralized control reduces the data management complexity compared to the distributed database and mirrored database approaches described below. Also, with the

centralized control security of the data can be managed at the one repository rather than several as described under plans discussed below.

However, the indirect access approach has the disadvantages of offering a single point of failure (with one repository), complexity in linking remote distance learning providers (several are depicted in Figure 18) to the CDSAR database for necessary access, complexity and redundancy for distance learning content providers in acting as a proxy between students and ECI, and a higher traffic volume for the one centralized database server.
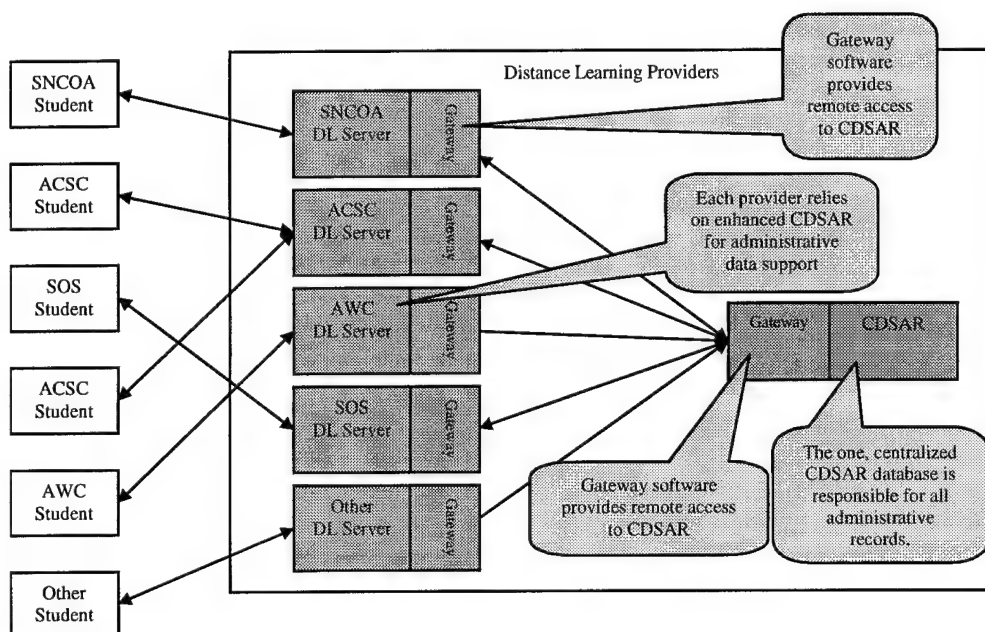
Figure 18. Indirect access approach showing multiple students and distance learning providers.[19]

- **Enhanced CDSAR with Direct Access**

In this approach the students are consumers of both ACSC distance learning course content and CDSAR student data content. (See Figure 19.) ACSC acts as both provider of course content to the consumers and consumer of CDSAR student data. In this approach, however, students (consumers) interact directly with the student data content provider (ECI's

---

[19] Senior Non-commissioned Officer Academy (SNCOA), Squadron Officer School (SOS), and Air War College (AWC) are three other Air Force education providers that are depicted in this and the following drawings as distance learning providers.

CDSAR) rather than interacting with the distance learning content provider (ACSC). (Figure 20 depicts the relationship of the components in the direct access approach while Figure 21 depicts the configuration for multiple distance learning providers.) ECI as a student data content provider is, as a consequence, responsible for providing the interface features for direct access by both the distance learning providers and the students—in this case for both ACSC and its students.
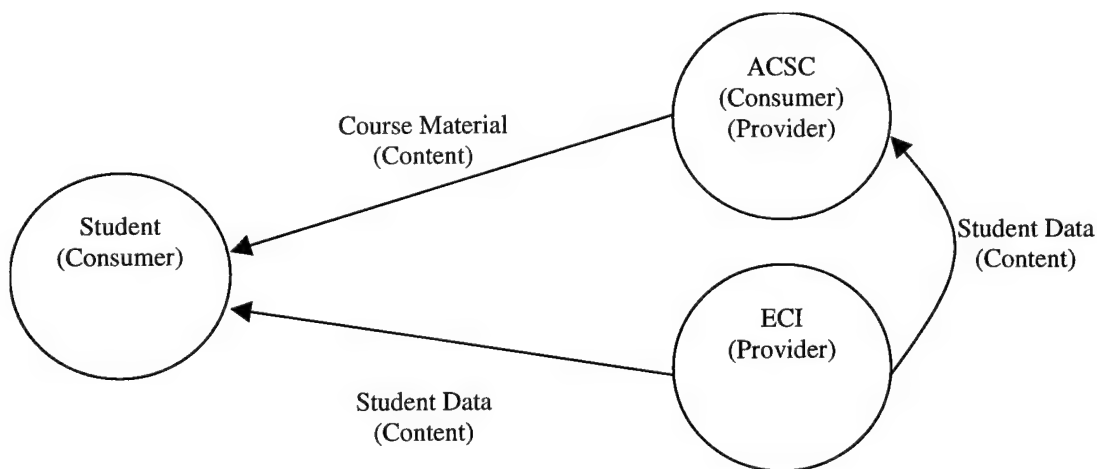
Figure 19. Consumers and providers for enhanced CDSAR with direct access.

The enhanced CDSAR with direct access offers many of the same advantages and disadvantages as the indirect access approach. However, allowing direct access by a larger base of consumers (the student population) introduces greater security risks as well as risks to the overall integrity and validity of the data. Also, additional resources would be required by ECI to support both the distance learning providers and students as direct consumers of the student data content. Figure 20 and Figure 21 depict the relationships of the components in the system and the support for multiple distance learning providers, respectively.
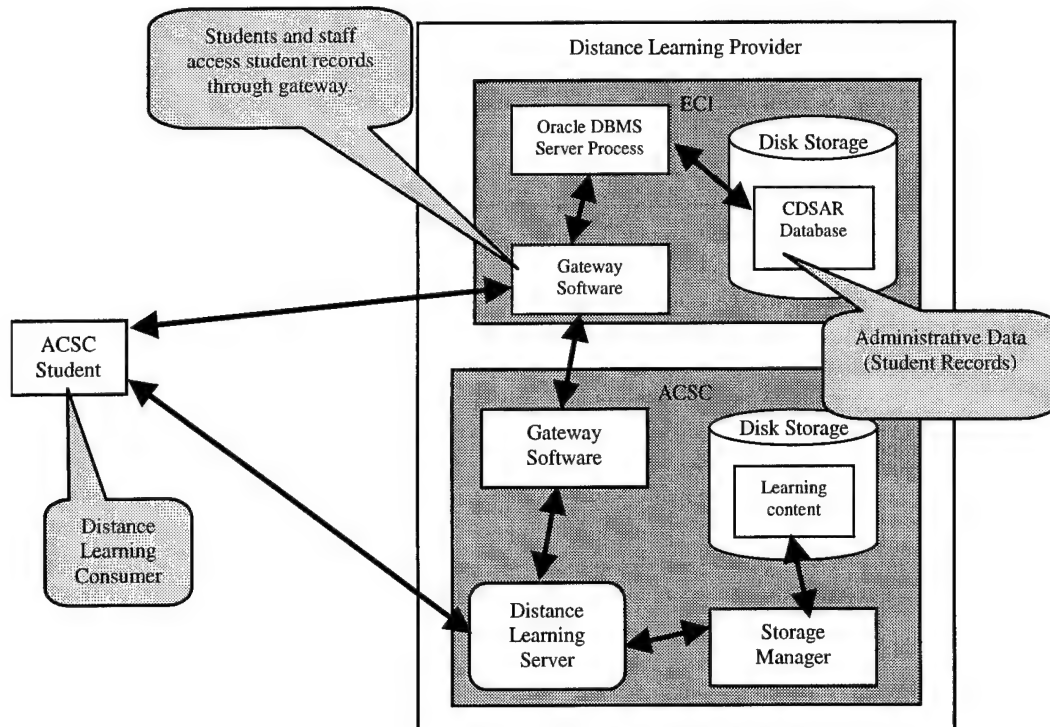
Figure 20. Relationships of the entities and software components for direct access.
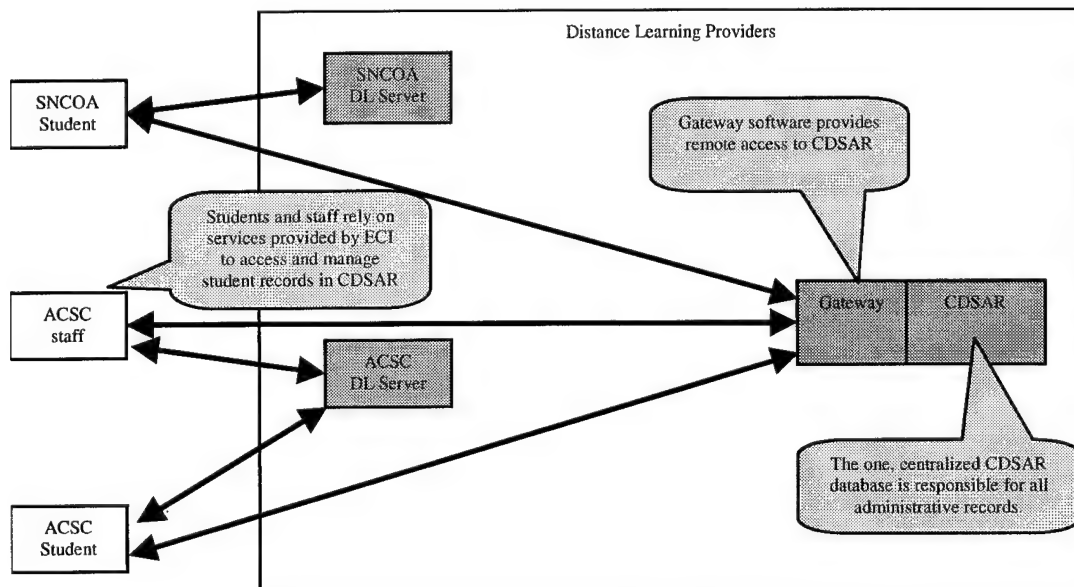
Figure 21. Using direct access for multiple distance learning providers.

- Distributed CDSAR

In the distributed CDSAR approach, each distance learning content provider maintains the student data content relevant to their function in a local database (see Figure 22). Meanwhile, a global distributed DBMS manager links the local provider databases together along with additional data stored within CDSAR. Larson refers to this type of distributed system as a "loosely-coupled" distributed database management system [Larson95: 12-13].
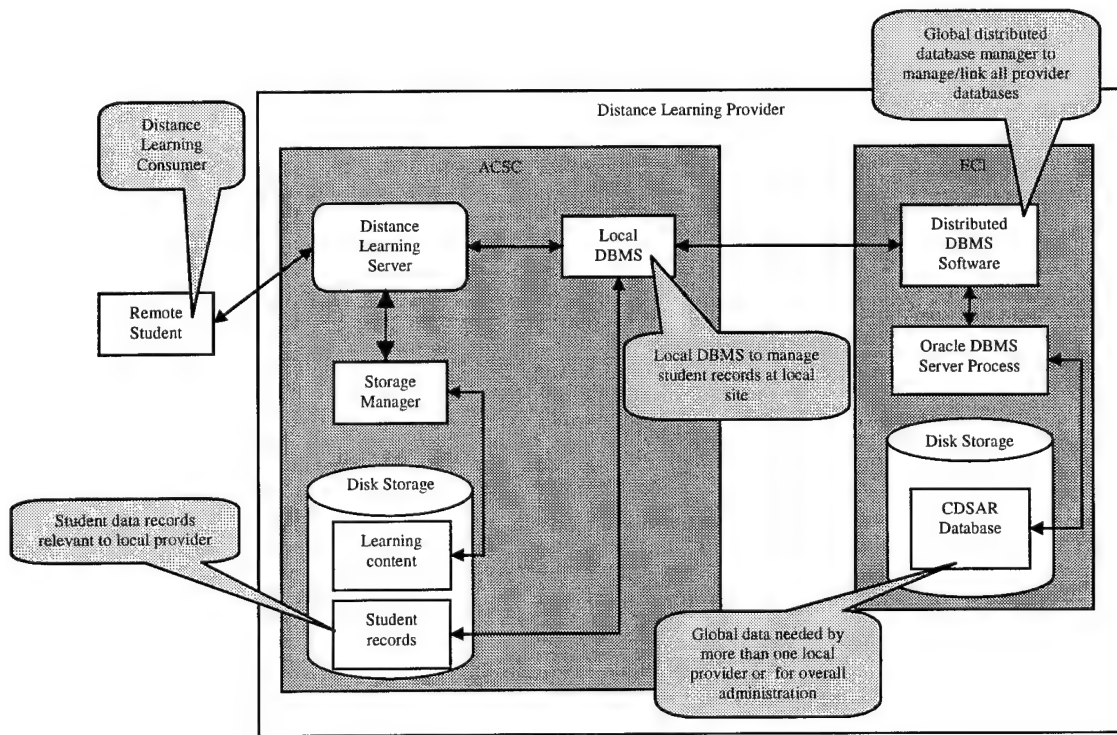
Figure 22. Relationship of entities and components of distributed approach.

The advantages of the distributed CDSAR approach include less workload for the CDSAR database server (because student data content requests will go to the distance learning provider), local autonomy for the distance learning content providers over their own data, and ease of integration of additional distance education providers into the larger system (because they will be able to apply similar technology and resources as existing distance learning providers). As Figure 23 shows, each distance learning provider would maintain their own database of the

subset of data relevant to their operation. Also, this distributed approach allows each distance education provider considerable flexibility in tailoring their systems to support student access to both the student data as well as course content.
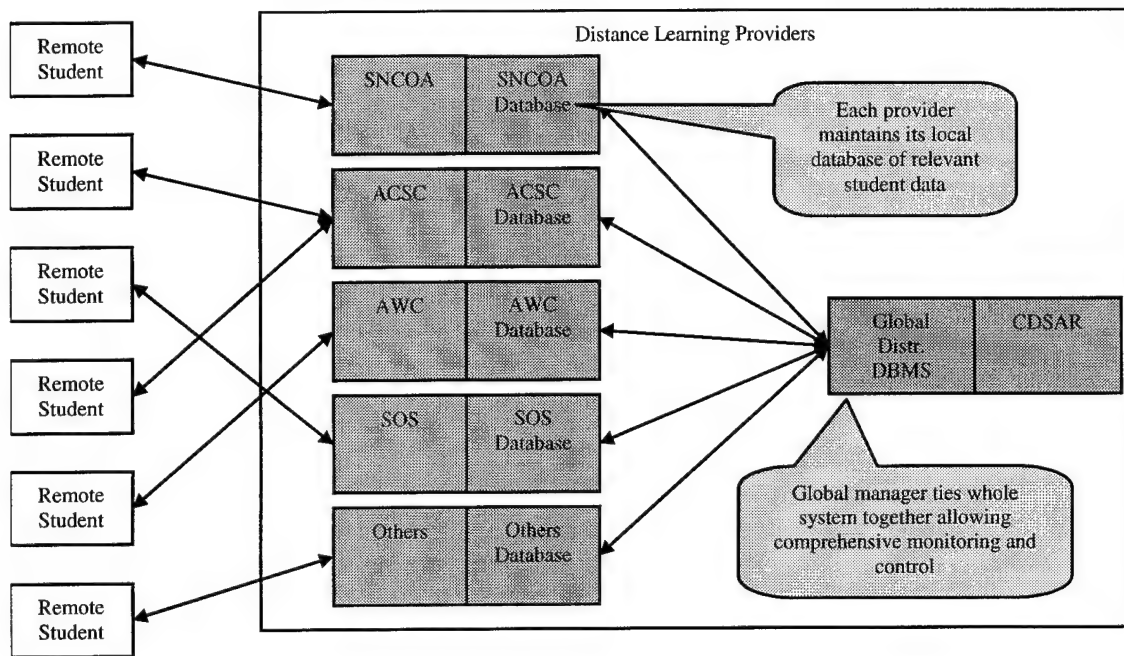
Figure 23. Distributed configuration being used by multiple distance learning providers.

One disadvantage of the distributed CDSAR approach is the possible complexity that may develop for the global data schema due to the differing data needs of the individual providers. Using a *homogeneous* distributed DBMS—one where all components are from the same software manufacturer or conform to the same standards—would reduce the risks associated with employing a complex distributed schema. Another disadvantage is the need for extensive coordination and cooperation among the various organizations to support the distributed architecture. Also, slow performance may result if a distance education provider requires non-collocated student data content.

- Mirrored CDSAR

    The mirrored CDSAR approach continues to treat ECI's CDSAR database as the central

repository for student data content. Meanwhile, individual distance education content providers

mirror the data relevant to their needs within a local database. This mirroring concept is also

known as *data replication*. (Korth and Silberschatz provide a discussion of replication issues for

DBMSs [Korth86: 408-418].) Figure 24 depicts the relationship of the components involved for

this mirrored approach, while Figure 25 shows the arrangement for multiple distance learning
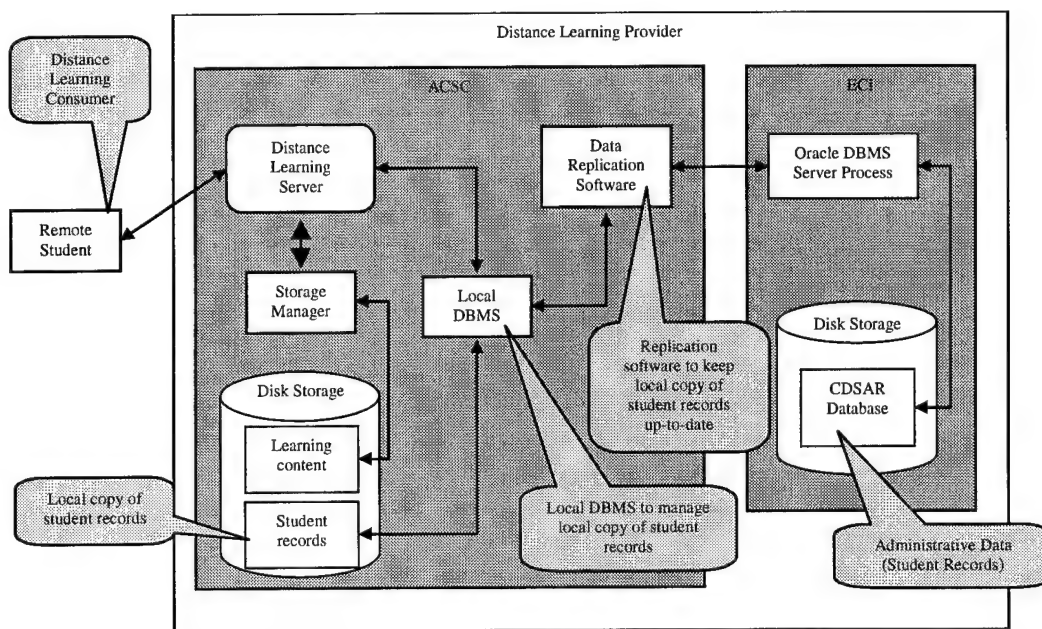
providers.

Figure 24. Relationship of components under mirrored CDSAR approach.

    The mirrored CDSAR approach offers the advantage of lower traffic at the CDSAR site

because only distance learning providers, not students, access the data, and that access would be

on a predictable, scheduled basis (to replicate the data). Meanwhile, security of the data in the

central repository would be more easily maintained since only the distance education providers

would interact directly with the central repository. This approach also offers the advantage of

allowing individual distance education providers to upgrade to the mirrored architecture at the

time of their own choosing since the central repository would continue as it currently exists. Finally, this approach would allow the distance education providers the greatest flexibility in designing and building applications that allow students to access and interact with both the course content and student data content.
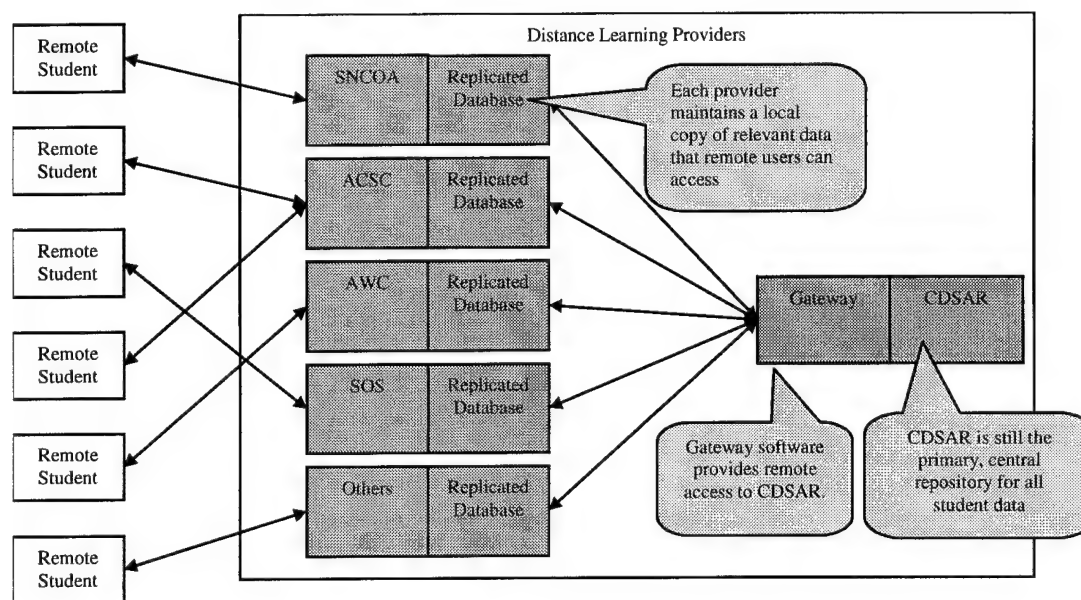
Figure 25. Mirrored CDSAR approach with multiple distance learning providers.

A significant disadvantage of the mirrored approach is that the data replication is one-way only: from CDSAR to the local DL providers[20]. Another disadvantage is the increase in data administration resource requirements at each distance education provider site; specifically, a duplication of data management skills and software resources would be necessary. Also, the expense and complexity of the replication architecture may prove greater than with the other approaches described.

- **No CDSAR**

The "No CDSAR" approach is the final alternative considered here. With this approach each distance education content provider independently and completely maintains the student

---

[20] Note that ACSC currently only has 'read' access to the data in CDSAR, so this represents no loss in capability.

data content for its needs. As shown in Figure 26 the current central repository, CDSAR, plays no role in this architecture.

One advantage of the no-CDSAR approach is that each distance education content provider has complete control of and access to the needed student data content. Meanwhile, performance should be better both for the distance education provider—because the data is collocated—and the students—because the distance education provider's database server should be less taxed than the CDSAR server. This also eliminates the expense associated with operating the centralized repository.
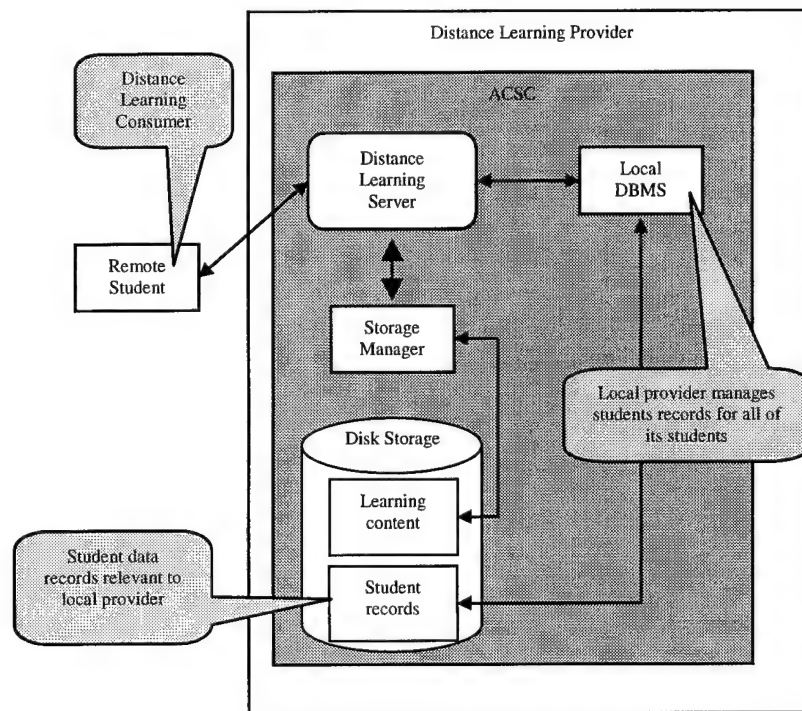
Figure 26. Relationship of components in no-CDSAR approach.

A disadvantage of the no-CDSAR approach is that each distance education provider must provide the resources to administer its own local database (see Figure 27) of student data content—i.e., the expense of operating the central repository is distributed among the various distance learning providers. Also, data will likely be stored and maintained redundantly if a student subscribes to more than one distance education provider (e.g., enrolls in courses from two

or more providers). And generating reports based on students of more than one distance learning providers would be difficult because the data would be dispersed.
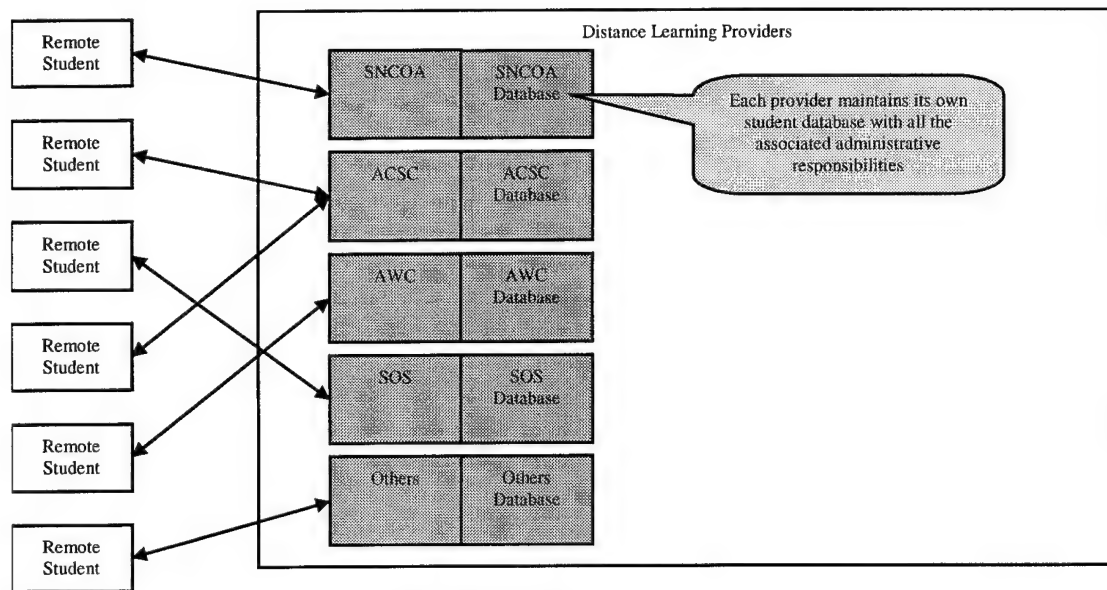
Figure 27. No-CDSAR approach with multiple distance learning providers.

3.4 Recommendations for Mirrored Approach for Legacy Data Access

A review of these alternative approaches to providing access to the student data content required by distance learning providers, such as ACSC, indicates that the mirrored-CDSAR approach is the most viable for the following reasons:

1. The existing legacy system, CDSAR, is least affected.

2. Distance learning content providers gain direct access to the locally stored, replicated data, allowing the distance learning staff to more readily integrate the student data content with other components of their systems.

Bukhres and Elmagarmid [Bukhres96: 39] describe the type of integration suggested in (2) above as *logical integration* because access is transparently provided to all databases through an interface that appears as one logical database. As such, the databases that comprise the system each maintain their local autonomy.

Further discussion of consumer versus provider will reflect the distance learning provider (ACSC) as the consumer of student data content and ECI as the provider of student data content to each distance learning provider; this is depicted as the region of Figure 28 enclosed by the dashed line. The remainder of the discussion within this thesis explores the feasibility of using a small-scale software system to provide data replication under a mirrored-CDSAR approach and, thus, allow ECI to provide student data content more readily to each student data consumer.
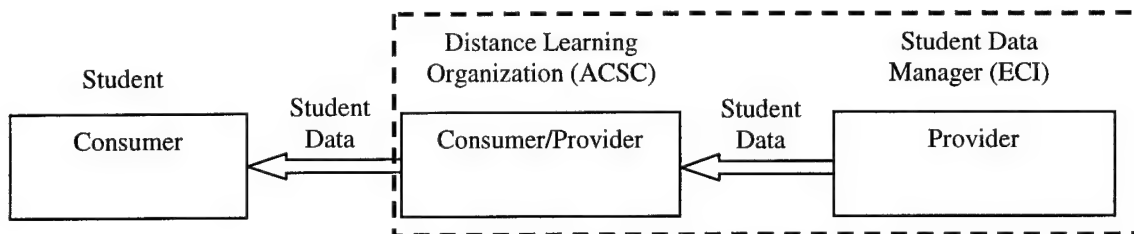
Figure 28. Providers and consumers for student data content.

3.4.1 Overview of Approaches to Replication

Several commercial RDBMS manufacturers already offer replication facilities for their products. However, these RDBMS products require a homogeneous environment—i.e., the central and remote databases must be maintained by RDBMSs from the same vendor. By requiring homogeneous systems, each RDBMS manufacturer is able to improve replication performance and features through proprietary techniques. An alternative approach to the proprietary, homogeneous DBMS solution is to employ methods that support access to a heterogeneous environment consisting of RDBMSs from different manufacturers. As already briefly discussed in Section 2.3.3.6, the application of JDBC using Java allows access to a variety of RDBMSs.

To investigate the feasibility of employing Java for the replication system, two significantly different client/server designs are considered. (Rauch [Rauch 96: 66-72] provides a discussion of issues surrounding client/server architectures, including advantages and

disadvantages of the two-tier and three-tier design models.) Both designs involve use of the Java programming language to access the remote databases via a TCP/IP network. The data is replicated from a source database located at one URL to a destination database located at a second URL. Both designs use the JDBC API for database access.

The first client/server design involves a two-tier model that requires replication processing be performed by the client process. Meanwhile, the second design employs a three-tier model using an CORBA ORB and a distributed Java object to place the majority of the replication processing in the server process. Benchmarks are gathered showing the relative performance of the two design approaches. The specific tests compare the time to complete the replication task given a homogeneous data set as well as a heterogeneous data set representative of real-world data from the CDSAR database.

3.4.2 Client-side Java/JDBC Two-tier Replication

The first design configuration employs a Java applet that is executed by the student data consumer (ACSC staff). JDBC API calls within the applet support connections to two relational databases through the use of the JDBC API and third-party JDBC network drivers. By designing the applet to establish connections to both the provider (source) and consumer (destination) databases, SQL statements can subsequently be executed to extract and transfer the desired records. The design is considered *two-tier* because the application logic is located in the client process—the first tier—which accesses data controlled by a server process—the second tier.

At primary issue with this two-tier version is the effect on performance of placing the replication processing within the client process. The steps that must be performed for the replication operation under this approach are depicted in Figure 29 and broken down as follows:

1. The applet sends a SELECT query to the source RDBMS to extract the data of interest.

2. The source RDBMS returns the result set for the query.

3. The applet performs any necessary processing on the data. (At a minimum, the language-dependent translation routines must prepare the data for the INSERT query in the next step.)

4. The applet transmits an INSERT statement with the necessary data to the destination RDBMS.
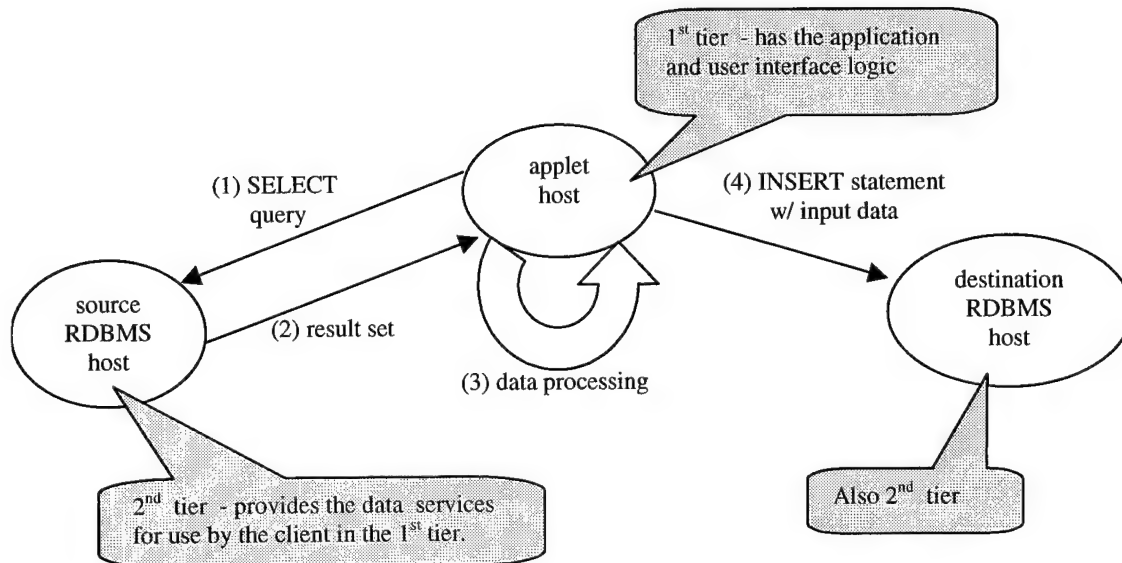
Figure 29. Diagram of activity for two-tier Java/JDBC processing.

Note that SQL "SELECT" queries into the source database produce result sets of relational records that must be transferred back to the applet (either all at once or in portions as needed, depending on the JDBC implementation), processed by the applet, and, subsequently, transferred to the destination database via an "INSERT" query. As a result of this approach, result set data must be transported among all three nodes of the supporting network. Meanwhile, additional processing (overhead) required to accomplish the replication operation takes place within the client.

3.4.3 Server-side Replication Using Distributed Objects in a Three-tier Architecture

The second design eliminates the applet client host as a node of travel for the replicated data and, meanwhile, takes advantage of the likely higher performance features of the provider's server host. This model employs a CORBA ORB and a distributed replication object to eliminate

the need to pass the data through the client applet for processing. A client object within the applet communicates with one or more distributed server objects that perform processing on another host computer[21]. The data is, as a consequence, transported between the source database platform and the destination with no additional detour through the client applet's host computer. This design is considered three-tier because only the user interface (presentation) features are located in the client applet—the first tier—while the application logic (replication) is located in a separate and distinct process—the second tier. The data server (RDBMS) also remains separate, as the third tier.

This model can be employed with the server (implementation) portion of the distributed object(s) located on a host computer at either the source or destination location. The general steps for completing the replication by use of distributed objects on the source are:

1. The applet remotely invokes the functionality of the distributed object on the source by accessing the object's client interface.

2. The server portion of the distributed object (on the host at the source location) performs a SELECT query on the source RDBMS to extract the data of interest.

3. The source RDBMS returns the result set for the query.

4. The server portion of the distributed object processes the data and prepares the INSERT statement.

5. The server portion of the distributed object transmits the INSERT statement with the necessary data directly to the destination RDBMS.

The diagrams in Figure 30 and Figure 31 depict the source vs. destination location versions, respectively, with the assumption that the server portion of the distributed object executes on the same host as an RDBMS. Since the applet, itself, is a downloaded program, it can be executed from any computer that has access to the TCP/IP network to which the database hosts have access.

---

[21] For the immediate discussion, assume this processing occurs on the same host computer as the database server. However, by employing CORBA distributed objects and network-capable JDBC drivers, the application logic may be located on any other TCP/IP network node that is reachable by the software components involved.
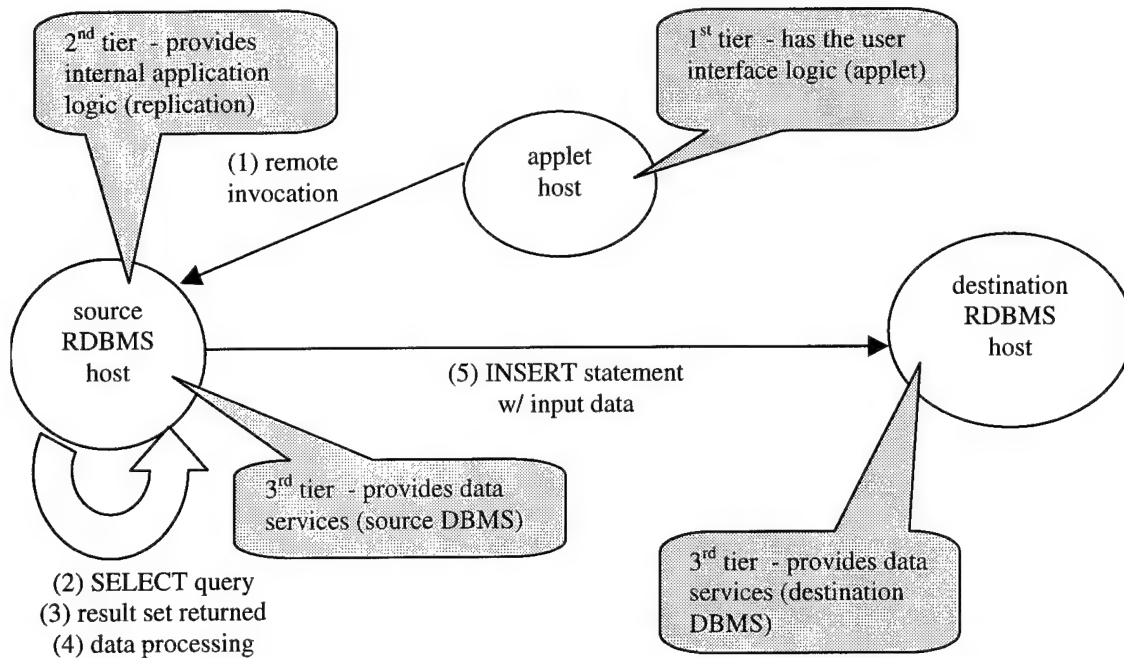
Figure 30. Diagram of activity with distributed object at source.
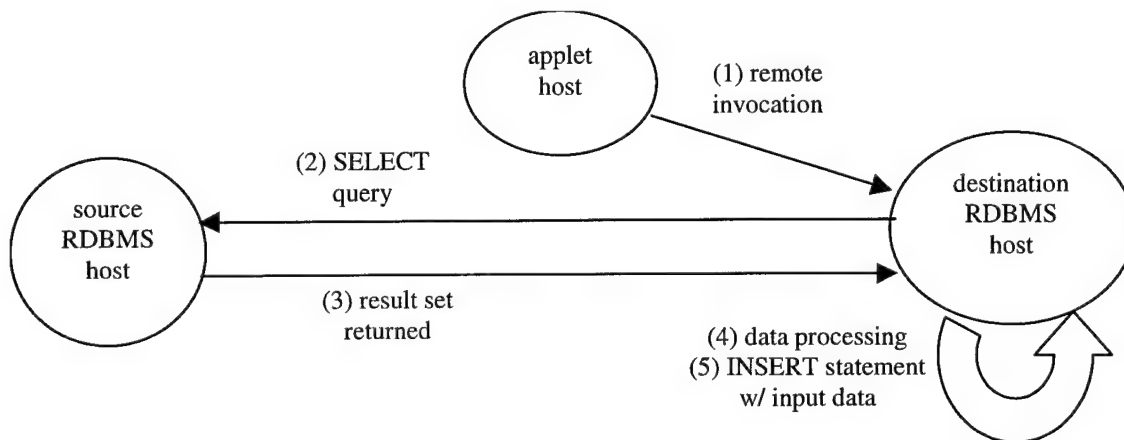
Figure 31. Diagram of activity with distributed object at destination.

3.5 Implementation and Testing Strategies

Chapter 4 will describe the details of implementing the two design configurations previously discussed to support the concept of replicating data from the CDSAR database at the ACSC location. Implementation of these two designs is intended to reveal:

1. Software and hardware design considerations for implementing the two-tier Java-only approach versus the three-tier approach using Java and CORBA.

2. Level of difficulty for implementation given the necessary design and programming steps involved.

3. Relative differences in replication time for the two configurations for several test cases. Note that only "wall time" for the replication as measured from the client will be recorded. This is due to (1) the Java API not providing a direct means to retrieve CPU processing time and (2) the replication being spread among (possibly) multiple computers. In the latter case, realizing that the entire operation is I/O-bound and that the significant measure of interest is round trip execution time reduces the value of knowing individual CPU execution times.

Chapter 5 provides the results from testing the completed implementations. Based on the results of the tests and other factors of the two architectures, one design is selected as the basis for a proof-of-concept system. The proof-of-concept system is used to demonstrate the capabilities and potential offered by the replication system to benefit operation of the ACSC distance learning program as well as support an enterprise model for providing student record content to other Air Force distance learning providers.

IV. Design and Implementation

This chapter presents the design, implementation, and test preparation for the two replication approaches introduced in Section 3.4. Issues surrounding design and implementation decisions are discussed where applicable. The recent introduction of many of the new software technologies (Java, JDBC, CORBA, IIOP) raised some development concerns that will also be addressed here and in Chapter 5. All Java source code written to support the implementations in this chapter and Chapter 5 may be obtained from the author. And a list of the commercial software packages employed during the design and implementation is provided in Appendix C.

4.1 JDBC driver issues

Both the two- and three-tier designs discussed below rely on the Java relational database access API called JDBC to support the mirrored-CDSAR model of Section 3.3. This section describes two key aspects of JDBC that are important to design and implementation but that have not yet been presented.

4.1.1 JDBC-ODBC Bridge

Related closely to the application of JDBC is that of ODBC currently used by many database application developers. While RDBMS vendors are developing upgrades or extensions to their products to allow direct JDBC access, current usage of JDBC relies on the support of the ODBC interface for RDBMS access. The ODBC interface is accessed through a piece of middleware known as a JDBC-to-ODBC bridge. This technique of using existing ODBC middleware has allowed for more rapid industry-wide deployment of Java and JDBC applications since access to RDBMSs is gained without requiring extensive RDBMS upgrades. A list of the four categories of JDBC drivers and a description of each can be found at the Javasoft web site [JDBC97]. Drivers implementing a JDBC-ODBC bridge make up the Type 1 category of JDBC drivers.

4.1.2 Networked JDBC Operation

Since several of the configurations that are examined under the two design approaches require communication across a TCP/IP network to each data source, the supporting JDBC driver must include the ability to transfer SQL queries and resulting data across a network. Note that this network capability is not a standard feature of the JDBC API, but one that is dependent upon each software vendor's implementation of the JDBC standard. Several commercial network-capable JDBC drivers are available, one of which is used in this study.

Both designs that are discussed below employ the JDBC API, a JDBC-to-ODBC bridge, and, as necessary, a network-capable JDBC driver to access the relational databases. The distinction in the use of JDBC in this study lies in whether the JDBC calls are made by the client process—under the two-tier model—or a distributed server process—under the three-tier model.

4.2 Two-tier, Java/JDBC Replication Applet

As discussed in Section 3.4.2, the two-tier version of the replication applet places all of the application logic on the client's host. This scenario requires that the applet establish connections to both the source and destination databases and transfer data through the client applet for processing before sending it on to the destination.

The diagram in Figure 32 depicts the software components of a system that uses a downloadable Java applet, a network-capable version of JDBC, and ODBC to interface with a database. The steps and components involved in the process of interacting with the database are as follows:

Phase One: Retrieve the Applet

1. The client platform web browser requests the Java applet from the web server.

2. The web server transmits the applet to the web browser.

3. The applet executes within the Java virtual machine of the web browser.

4. Within the applet, the JDBC driver manager is loaded to perform the role as translator to the specific JDBC driver.

5. The JDBC driver manager loads the specific JDBC driver that is capable of communicating with the JDBC data source. In the case of a network version, the JDBC driver must be designed to support network connection to a remote JDBC server that is on (in this example) the web server platform.
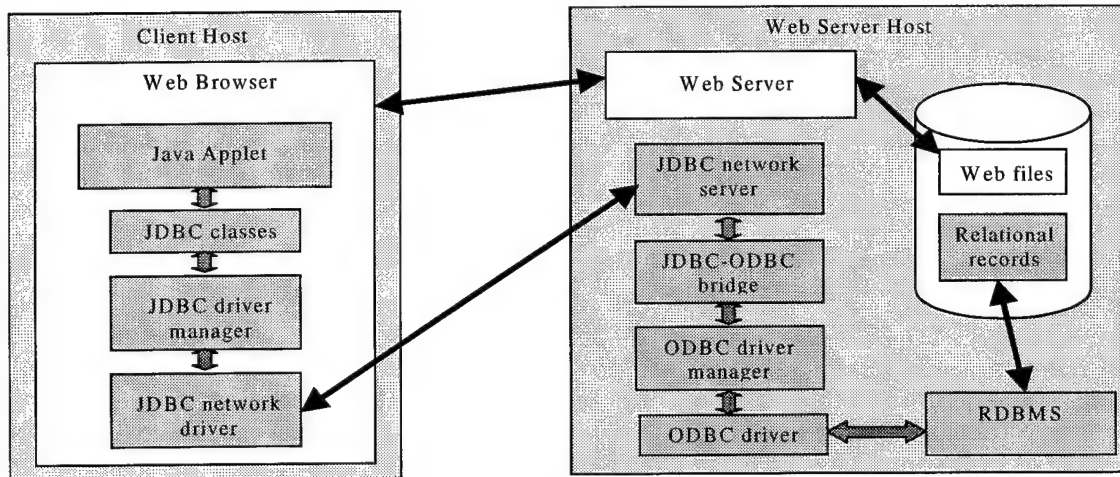
Figure 32. Software components of the two-tier Java JDBC applet approach.

6. The applet requests a connection, through the JDBC components, with a specific data source by specifying the precise URL identifying that data source location and name.

7. The JDBC connection request is passed to the JDBC driver manager, which in turn interfaces with the chosen driver.

8. The driver uses the URL to identify and connect with the JDBC network server on the web server platform.

9. The JDBC network server communicates through the JDBC-ODBC bridge with the ODBC driver manager, which in turn loads the ODBC driver associated with the data source. After the connection is established through the ODBC driver to the data source, the Java applet can proceed with SQL queries.

---

[22] A *data source object* in this context is any resource that stores data and is accessible via a JDBC driver using relational DBMS query syntax supported under the SQL92 standard. For all work done in this thesis that equates to any source of data accessible through an ODBC driver, or more specifically, a database created by a RDBMS for which an ODBC driver is provided and installed. In subsequent drawings, the ODBC components will be left out and their support is assumed when the JDBC driver uses a JDBC-ODBC bridge. Also, note that ODBC drivers are available that support ASCII text files (i.e., not DBMS-managed databases) as data sources; however, the driver for a text file database (or other simple DSO) offers only a minimal level of functionality compared to a more capable ODBC driver supporting databases created and maintained by a "full-fledged" RDBMS.

<u>Phase Three: Execute the SQL Statements</u>

10. The Java applet prepares SQL statements that, subsequently, travel through the JDBC driver manager to the JDBC driver. The JDBC driver translates the calls into a data format suitable for transport across the network to the JDBC network server.

11. The JDBC network server translates the network data back to JDBC calls and passes those to the JDBC-ODBC bridge.

12. The JDBC-ODBC bridge translates the JDBC calls into ODBC calls and passes those through the ODBC driver manager to the ODBC driver.

13. The ODBC driver executes the SQL statements embedded within the ODBC calls on the associated database.

14. If results or exceptions are generated from the execution of the SQL on the database, the data is sent back to the Java applet through the reverse steps of the process already described.

In the replication process, the steps outlined above are applied to both the source and the destination database. After the connection is established to both databases, SELECT queries are performed on the source database to extract the necessary records. The applet processes the result set from the query and translates the data into suitable INSERT statements. The applet then transmits the INSERT statements to the destination database. Database metadata and query result sets are extracted as necessary to assist in processing the data.

- Initial Design Using ODBC-encapsulating Classes

The first implementation of the two-tier design used a commercial Java database connectivity package developed by XDB Systems, Inc., called JetConnect. The JetConnect system employs specialized classes developed to encapsulate and emulate the database connection and query steps performed for ODBC-based applications. The system also includes client and server classes that allow communication across a TCP/IP network to existing ODBC data sources. Note that while supporting many of the same features as the JDBC specification, the JetConnect classes are intended as a proprietary alternative to, rather than an implementation of, the JDBC specification.

The design and implementation of the experimental applet employing JetConnect included:

- Development of a set of user interface classes to support connection to a data source,

- Development of a class representing a predefined set of query operations that would extract data from the first DSO and insert it into the second DSO,

- Development of an applet class to integrate the connection and query classes,

- Creation of an HTML file in which to embed the applet and specify default execution parameters

For testing purposes, Microsoft Access95 was used to create source and destination databases. ODBC database sources were then established for the databases using the Microsoft Windows95 *ODBC Data Source Administrator* and the Microsoft Access95 ODBC driver. The JDBC network server software, a companion XDB product called *JetPort*, was executed on both the source and destination host computers.

The applet was executed through the *appletviewer* utility program provided with the Sun Java Development Kit. The appletviewer program supports overriding the Java virtual machine default security features that limit applets to only communicating with the computer from which they are downloaded. Therefore, the applet was able to connect to any designated (and reachable) computer, which would not have been possible with the Java virtual machines of current popular web browsers.

The design successfully accomplished the goal of retrieving the contents (via a single query) of one database and inserting it into another. The next step was to accomplish the same functionality using the Java version 1.1 JDBC API.

- **JDBC Version of Initial Applet**

After successfully employing the JetConnect API to accomplish the extraction and insertion tasks between two databases, the next step was to convert the applet to the non-proprietary, JDBC implementation. All the functional components discussed for the JetConnect

73

version above still applied. However, rather than using the JetConnect classes that encapsulate the ODBC functionality, only classes from the JDBC API (otherwise known as the "java.sql" package of the Java core API) were used in the design of the applet.

The JDBC driver used to support network communication was provided by XDB as a separate component of the JetConnect software package. XDB's implementation of the JDBC specification incorporates a JDBC driver that translates the JDBC calls to their JetConnect equivalents, allowing use of the same client and server network communication classes employed in the non-JDBC, JetConnect version above. XDB's driver falls in the categories of both Type 1 and Type 3 JDBC drivers [JDBC97] since it incorporates both a JDBC-ODBC bridge (used on the server side) and a DBMS-independent network protocol.

The implementation of the JDBC version proved somewhat more difficult for two reasons. First, the mapping between JDBC data types, SQL data types, and Access data types was not clearly defined in the available documentation. Some experimentation was required to correctly identify which JDBC and SQL data types matched the actual data types for the fields within the Access database.

Secondly, initial testing of the functioning applet proved successful when the data sources were both located on the same computer–i.e., the URLs differed only by the name of the ODBC DSO. However, testing of the applet with the source and destination DSOs on two different computers failed. After extensive troubleshooting and contact with XDB Systems, it was discovered that a "bug" existed in the XDB JDBC driver that only allowed concurrent connections to the same computer. XDB, subsequently, released a patch that corrected the problem and allowed concurrent connections to multiple computers.

- Expanding the Applet for Database Replication

The next phase of development involved providing the functionality that would perform the replication process on any number of tables between any two databases. This required adding the ability to read in the table schemas for the databases, parsing out the necessary table definition

74

information (including data types), and constructing the necessary queries "on-the-fly". The pseudo-code algorithm for the replication operation is listed in Figure 33.

Retrieve the destination (or source) database metadata

Extract the table names from the metadata

Delete the contents of the destination database

Repeat for each table name

    Build an SQL SELECT statement to retrieve the source data

    Build an SQL INSERT prepared statement to insert data into the destination database

    Store the data type information for each field in the table

    Execute the source query

    Repeat for each record retrieved from source query execution

        Repeat for each field in the record

            Copy the field value (using the stored data type info) into the INSERT statement

        Execute the INSERT statement on the destination database

Figure 33. Pseudo-code for replication algorithm.

The flexibility designed into the test program allows supports for any of the eight configurations listed in Table 1 relating number of platforms, JDBC driver options, and JDBC driver selection. In any version, using the Sun JDBC-ODBC bridge provided with the JDK 1.1 offers the faster performance (over the network-capable JDBC drivers) due to not having the network server middleware. However, this limits the applet or application to being located on the same host computer as the DSO. Alternatively, use of a network-capable JDBC server offers considerable flexibility for access to non-local DSOs at a cost in performance when accessing local DSOs. The three diagrams of Figure 34, Figure 35, and Figure 36 depict the three most likely configurations that would be applied for the scenarios of one, two, and three individual host computers, respectively. (Note that the number of hosts doesn't take into account the possibility that the DSO represents a database managed by a DBMS that itself has distributed files and components.)

Table 1. Configurations for hosts and JDBC drivers under two-tier design.

Number of Host Computers	JDBC drivers	DSO Accessibility
One	Sun JDBC-ODBC bridge only	Source DSO: Sun bridge Dest. DSO: Sun bridge
	1 Sun JDBC ODBC bridge 1 Network JDBC server	Source DSO: Sun bridge Dest. DSO: Network server
		Source DSO: Network server Dest. DSO: Sun bridge
	2 Network JDBC servers	Source DSO: Network server Dest. DSO: Network server
Two	1 Sun JDBC ODBC bridge 1 Network JDBC server	Source DSO: Sun bridge Dest. DSO: Network server
		Source DSO: Network server Dest. DSO: Sun bridge
	2 Network JDBC servers	Source DSO: Network server Dest. DSO: Network server
Three	2 Network JDBC servers	Source DSO: Network server Dest. DSO: Network server

Figure 34 shows a high-level view of the one-host version using only the Sun JDBC-ODBC bridge. Communication delays between processes in this version will be minimal. However, extra I/O transfer delays may be introduced due to disk "thrashing" from attempts to read and right consecutively from the source and to the destination databases if those databases are located on the same physical disk device.
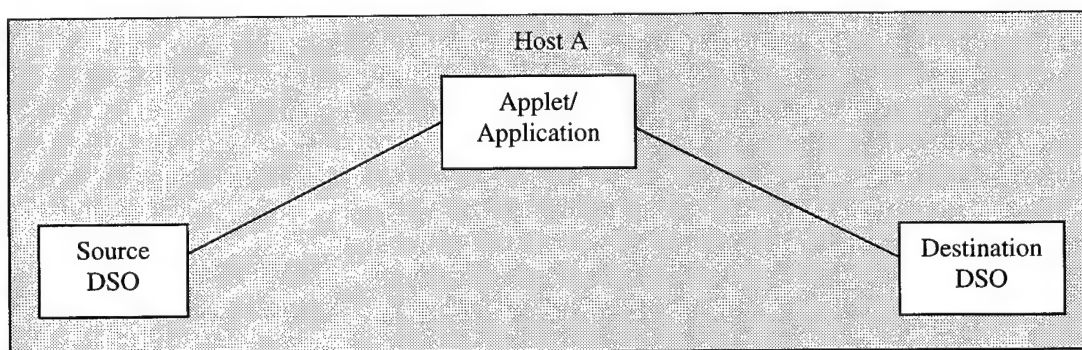
Figure 34. One-host version of two-tier design.

Figure 35 depicts the high-level view of the two-host version with the applet or application residing on the destination host computer. Only one network-capable JDBC driver is required. The move to two hosts will eliminate the disk thrashing now that the DSO files do not

reside on the same disk device. (This assumes the DSO files are not located on the same drive as part of a common network file system). However, there is the potential for reduction in performance due to lower bandwidth communication between the two host computers.
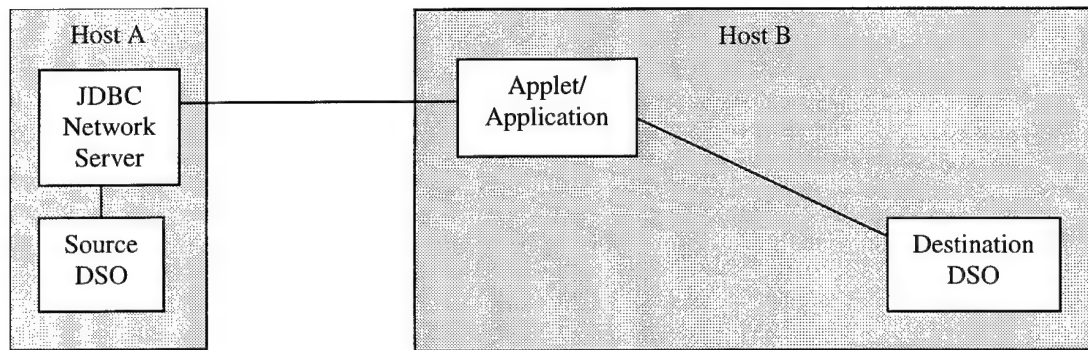
Figure 35. Two-host version of two-tier design.

The three-host version, depicted in Figure 36, allows the controlling applet to exist on a separate host from either the source or the destination DBMS. As a result, the three-host version requires network-capable JDBC drivers to access both DBMSs. The effects of relying on a lower bandwidth network will also likely reduce performance here. However, the disk activity for each host will be reduced to only that required for accessing a single DSO, thus eliminating the disk thrashing discussed under the one-host version.
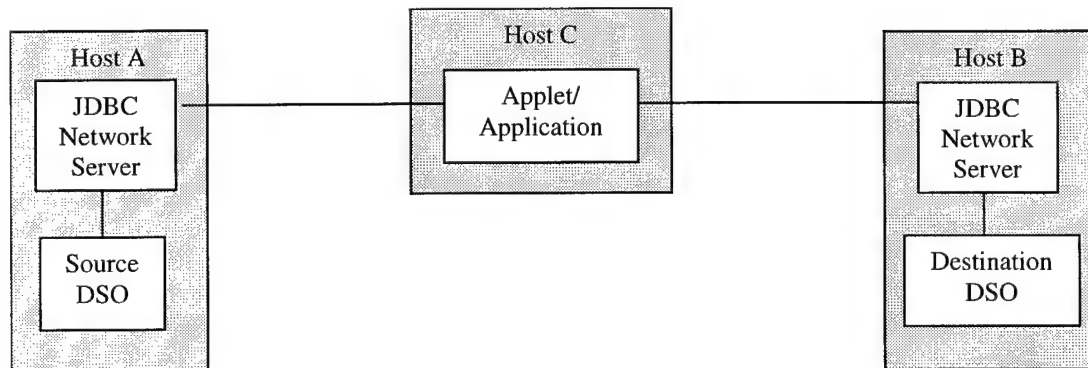
Figure 36. Three-host version of two-tier design.

For testing purposes, a command line Java application was constructed as an automated approach to creating the tables in the test databases, and an e-mail class was incorporated into the

replication applet to automatically send a results log to a specified e-mail address[23]. Also, table schemas and example data provided by ECI from the CDSAR database were used in the initial testing.

The replication applet functioned correctly for the defined schemas. However, interaction with the database located on the 33MHz 486 computer was perceptibly slow for even the small data sets involved. (A more detailed discussion of performance is provided later in this chapter and in Chapter 5.)

- Completing Development of the Test Program

Initial development only supported copying the data of two specific SQL data types—*integer* and *char*—in order to minimize the effects of incompatible or inconsistently implemented data types among the source (Microsoft Access) and destination (Microsoft SQL Server) DBMSs. Research proceeded with the creation of a Java/JDBC utility program to query each DBMS for metadata describing its data type support. The mapping of recognized SQL data types to those supported by the Access and SQL Server ODBC drivers is summarized in Table 2. Functionality to copy all SQL data types (except for *longvarbinary*[24]) from the first column of Table 2 was incorporated into the *DatabaseReplicator* class—the class encapsulating the replication algorithm. Note, however, that the two DBMSs do not support the identical set of data types, so replication by direct mapping (the approach taken in this study) practically limited the replication to only the common set of data types.[25] For example, SQL Server supports the 28-digit *decimal* type while Access does not.

---

[23] The e-mail log approach was chosen because security restrictions on applets loaded from a remote web server will not allow writing the log file to the local hard disk.

[24] The *longvarbinary* data type required the use of streams to transfer data from and to the data source and was not an item of interest for this study.

[25] If the data values from the source DBMS are of a type that does not directly map to a type in the destination DBMS, it may still be possible to copy to an "appropriate" data type with additional semantic and/or domain information.

Table 2. SQL type support from ODBC drivers.

| SQL Data Type | Type Support from RDBMS ODBC Driver | |
	MS Access ODBC Driver	MS SQL Server ODBC Driver
char	Yes; limited to 255 char	Yes; limited to 255 char
varchar	Yes; limited to 255 char	Yes; limited to 255 char
longvarchar	No	Yes; generates ODBC error over 255 char
numeric	Yes; called 'currency'; 19 digits precision	Yes; 28 digits precision
decimal	No	Yes; 28 digits precision
bit	Yes	Yes
tinyint	No	Yes; 3 digits precision
smallint	Yes; called 'short' ;5 digits precision	Yes; 5 digits precision
integer	Yes; called 'long'; 10 digits precision	Yes; called 'int'; 10 digits precision
bigint	No	No
real	No	Yes
double	Yes; 15 digits precision	No
float	Yes; but not listed in metadata	Yes; 15 digits precision
binary	Yes, but not tested due to unknown error in replication	Yes, but not tested due to unknown error in replication
varbinary	Yes, but not tested due to unknown error in replication	Yes, but not tested due to unknown error in replication
longvarbinary	Yes, but not tested due to use of streams; called 'longbinary'	Yes, but not tested due to use of streams; called 'image'
date	Yes, a.k.a. time, 'datetime' or 'timestamp'	No
time	Yes, a.k.a. date, 'datetime' or 'timestamp'	No
timestamp	Yes, a.k.a. time, date, or 'datetime'	Yes; but known only as 'datetime'; 'timestamp' has another meaning/purpose

After further analysis of the role of the replication class, two further enhancements were made. First, to support simplified use of the replication class, the functionality to read the database schema from a file was removed and replaced with support for extracting the schema information directly from either the source or destination database at run-time. For this schema extraction approach to work, the schema of the database that will be used to define the replication must be a subset of the other database schema. In other words, if the destination database is chosen to define the replication schema, then the destination database schema must be a subset of the source database schema. Conversely, if the source database is chosen to define the replication schema, then the source database schema must be a subset of the destination schema.

Second, despite the limitations previously discussed about logging to file for remotely loaded applets, support for file logging was added to the client-applet to support saving test results. The applet saves the results of all testing in a comma-delimited ASCII text file to allow subsequent importing into other programs for analysis. Support for multiple test cases was also

added to the client to allow unattended test operation, as well as the ability to specify the type of test.

The user-defined classes created to support the two-tier replication system are depicted in Figure 37, along with several core classes from which they inherit. (The attributes and aggregate components are not depicted in the diagram due to the high number involved.) The high-level interaction of the client classes is depicted in Figure 38, showing the method calls the client makes to the other objects in the program. Figure 39 shows a screen snapshot of the replication test applet as executed by the web browser. The Java source code for the two-tier *DatabaseReplicator* class as well as the other classes directly supporting database operations can be found in the Java source code package *macker.database*. The source code for the applet client can be found in the Java source code package *macker.mirror2tier*, with additional support classes located in package *macker.util*.
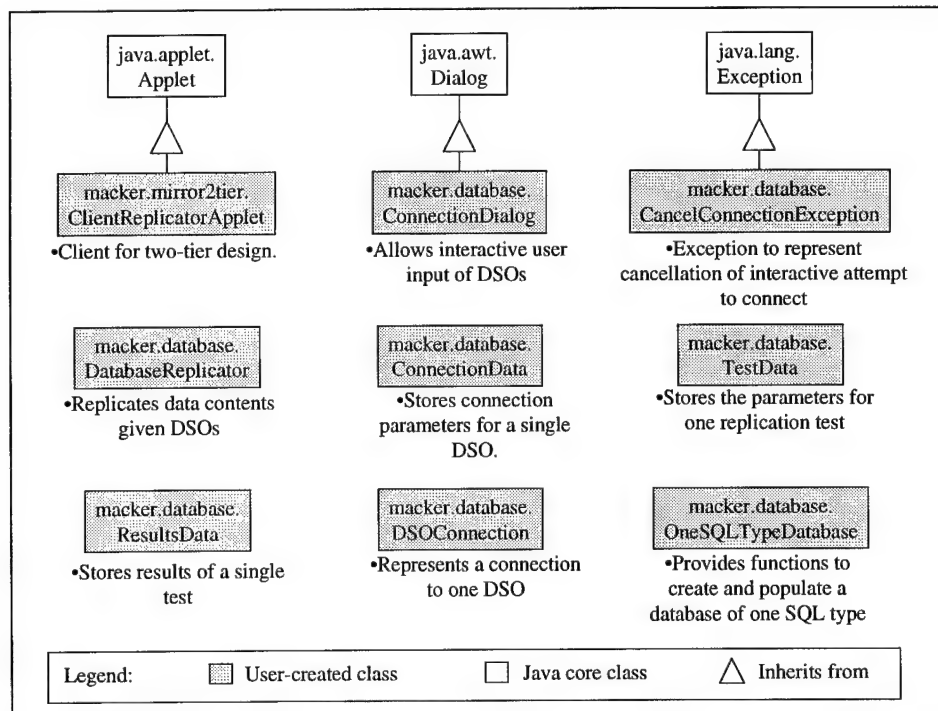
Figure 37. User-defined classes for two-tier design.
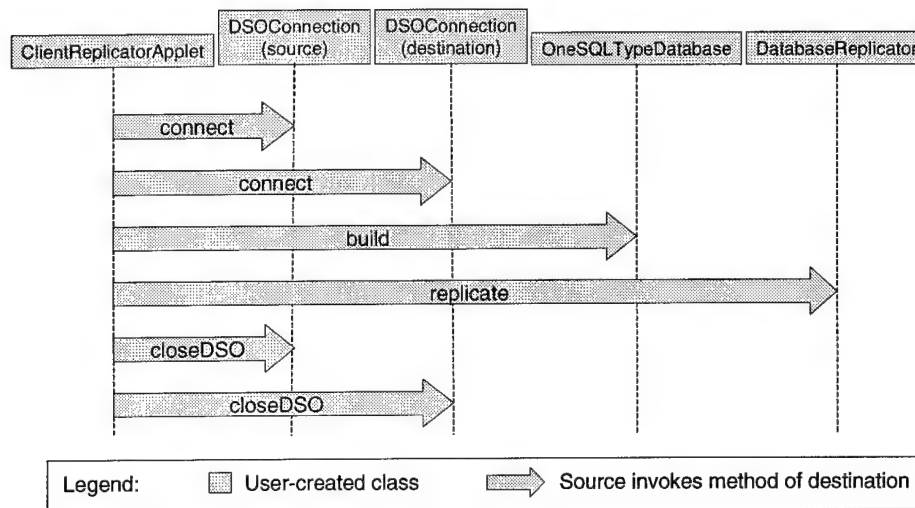
Figure 38. Client interaction diagram for two-tier client.
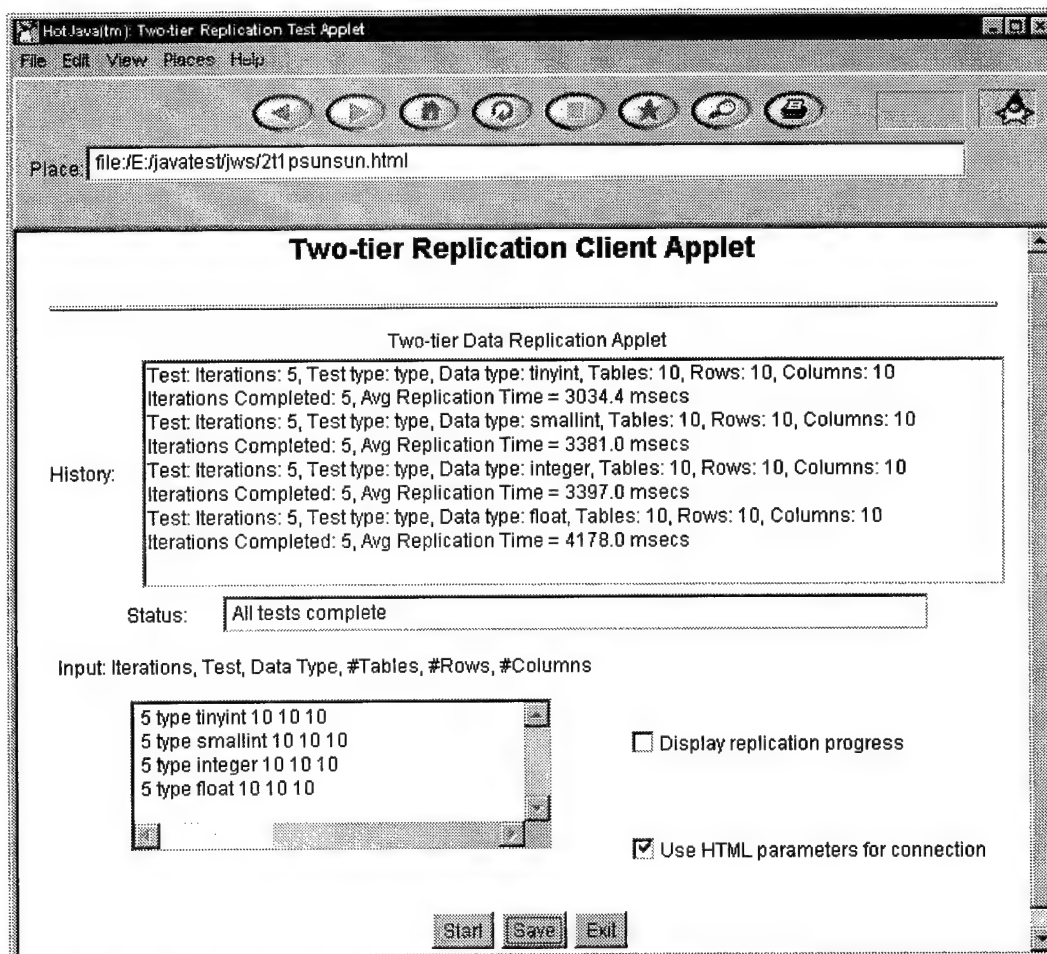
Figure 39. Screen snapshot of test replication applet.

81

4.3 Three-tier Java/CORBA/JDBC Replication Applet

To support the three-tier approach, selection of a CORBA ORB along with its associated development environment was necessary. The decision was made to work with an evaluation version of Visigenic Visibroker for Java for the Windows NT platform. Some design and implementation decisions were based on requirements introduced by the use of this specific ORB product. However, the details for implementation using the Visigenic ORB should be similar to those when employing any other CORBA 2.0-compliant ORB product.

4.3.1 Incorporation of CORBA Features

The design of the distributed client and server versions was based on the interface and object implementation both being statically defined (see Section 2.3.2.3 for a discussion of static versus dynamic CORBA interfaces). As such, the name of the distributed object can be referenced by the client at compile time. Also of note, since under the three-tier configuration all interaction directly with each DBMS is to take place in the middle tier (server portion of distributed object), no JDBC API calls that result in interaction with the database are allowed in the applet client. In other words, only the server object is allowed to directly interact with the DSOs.

Other notable enhancements or modifications to support the three-tier CORBA configuration included:

- **Creation of the *connect* method.** The two-tier version could create and pass a JDBC Connection object (representing a connection to a DSO) between its client application and the DatabaseReplicator. Because the client in the three-tier version was to have no interaction with the DSO, the connect method was created to allow the client to request that the object implementation establish the DSO connections.

- **Creation of the *disconnect* method.** The *disconnect* method was created to allow the client to request that the object implementation disconnect from the DSO.

- **Creation of the *initDatabases* method.** The two-tier version had initialized the databases with homogeneous schemas and test data directly from the client (as defined by the test cases). However, under the three-tier configuration, the client was to have no interaction with the DSO. Therefore, the *initDatabases* method

82

provides the interface for the client to request that the databases be initialized with homogeneous data sets.

Besides incorporating the modifications listed above, conversion to the distributed, three-tier version required performing the steps to create a statically-defined distributed CORBA object. Actual steps for designing and implementing the CORBA distributed version of the *DatabaseReplicator* class were as follows:

1. **Define the interface.** Identify all necessary interface features the distributed DatabaseReplicator class needed to offer. This primarily meant translating the public methods and exceptions into the corresponding IDL syntax. (Orfali and Harkey [Orfali97: 361-382] provide an overview of IDL and its mapping to Java.) Of course, the public constructors were not translated because the client is not permitted to instantiate a DatabaseReplicator object; instead the server object (see steps 5 and 7, below) instantiates the object implementation. Also, the new public methods identified above were incorporated.

2. **Create the IDL file.** Create the IDL text file containing the DatabaseReplicator class interface definition. Figure 40 shows the IDL used to create the DatabaseReplicator class. The module was given the name CORBA to differentiate it from the other already-created non-CORBA version of the DatabaseReplicator class[26]. The exception and methods are self-explanatory.

3. **Compile the IDL file.** Compile the IDL file using the development environment's IDL-to-Java compiler. This generates the stub and skeleton classes necessary to support the features of the interface defined by the IDL. The 'CORBA' module was translated to a Java package, and all generated classes were assigned to that 'CORBA' package. Helper classes were also automatically created that, among other responsibilities, provide methods for the client to bind to the object and transfer parameter values between the between the client and the object implementation.

4. **Write the object implementation code.** Create the code, using the Java language, that implements the features specified by the object's IDL-defined interface. Also, create any supporting code (for methods, attributes, exceptions, etc.) within the class files that are not intended to be visible to clients through the object's interface.

5. **Write the server object code.** The server object is used to instantiate the object implementation in the server environment, register the object with a local ORB, and prepare the object to await requests from clients.

6. **Compile all Java files.** Compile the object implementation, server, and helper class files to produce the associated Java bytecode class files.

---

[26] Note that the module could have been given any name that conform to the rules for identifier names and does not conflict with an IDL reserved word or an existing module name in the current namespace.

7. **Execute the server class.** Execute the server class using the Java interpreter on the execution platform. This step makes the distributed object and its services available for use by any other object that has direct or indirect access to the ORB with which the object is registered.

```
// DatabaseReplicator.idl
module CORBA
{
  exception ReplicationFailureException
  {
    string reason;
  };
  interface DatabaseReplicator
  {
    boolean connect(
       in string sourceJDBCDriver,
       in string source_url_dso,
       in string source_uid,
       in string source_pwd,
       in string destJDBCDriver,
       in string dest_url_dso,
       in string dest_uid,
       in string dest_pwd);
    boolean disconnect();
    void initDatabases(
       in string    dataType,
       in unsigned long  numberOfTables,
       in unsigned long  numberOfRows,
       in unsigned long  numberOfColumns);
    void replicate(in string schemaSource)
       raises (::CORBA::ReplicationFailureException);
  };
};
```

Figure 40. IDL for CORBA version of *DatabaseReplicator*.

4.3.2 Converting the Client Applet

Overall, the conversion of the two-tier client applet to the three-tier CORBA version required little modification. The primary changes were (1) including the code to bind to the distributed *DatabaseReplicator* object rather than instantiate a local version of the non-distributed object and (2) updating the references for connecting and disconnecting from the databases and initializing the homogeneous test databases to use the *DatabaseReplicator* class. Figure 41 shows the resulting classes that provided key support for the three-tier design. Figure 42 shows the high-level interaction diagram for the client. Note that in the interaction diagram for the three-tier client, after locating and establishing the connection to the distributed object (using the *bind* method of the *DatabaseReplicatorHelper* class), the client only interacts with the *DatabaseReplicator* object. Compare this interaction to that of the two-tier client, which must

84

also interact with the two data sources objects and the database-creation object, *OneSQLTypeDatabase*.
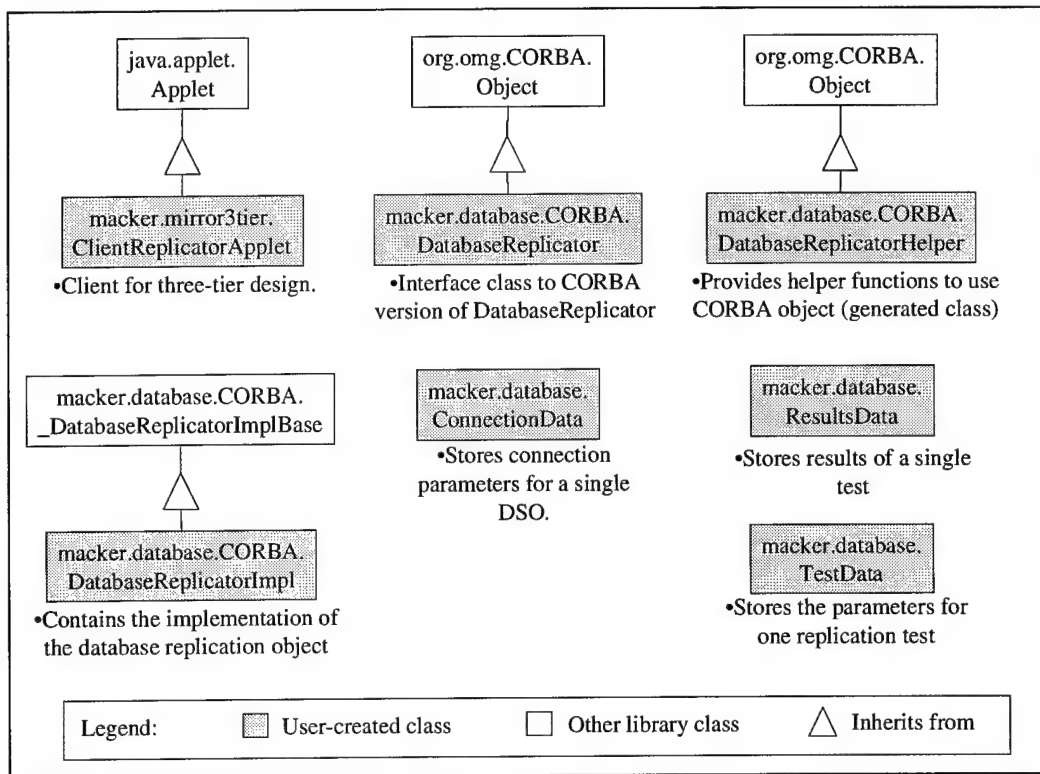
Figure 41. User-defined classes for three-tier client.

The Java source code for the CORBA version of the *DatabaseReplicator* class is located in the Java package *macker.database.CORBA*. Additional database operation support classes are located in the Java package *macker.database*. The client applet code is located in package *macker.mirror3tier*, with the code for additional support classes located in *macker.util*.

4.3.3 Three-tier Host Configurations

Under the three-tier configuration using CORBA, the number of separate hosts upon which the components can be distributed is as low as one and as high as six. Figure 43 shows these software components and the communication paths between them[27]. As under the two-tier

---

[27] This approach might also be called an 'n-tier' system when considering all the supporting processes and possible ways of distributing them among hosts. However, the application design remains three-tier, with the client providing the user interface, the server object providing the application logic, and the DSOs providing the data.

model, this diagram doesn't take into account the possibility that the DSO represents a database managed by a DBMS that has its own server components and files distributed. Note that some components, while allowed to be on separate platforms, must share the same local area network—e.g., a client application must be able to locate an ORB agent on the same LAN.
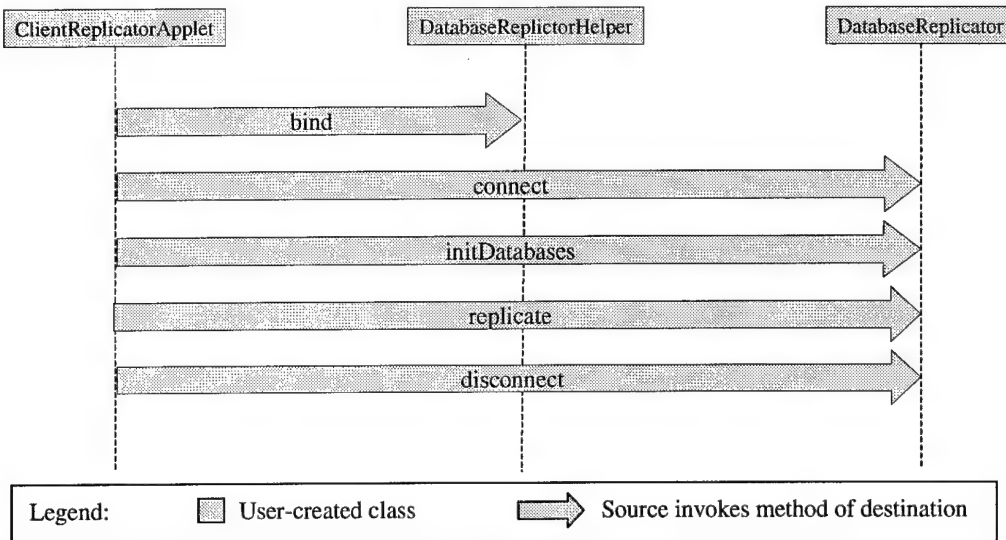
Figure 42. Client interaction diagram for three-tier design.
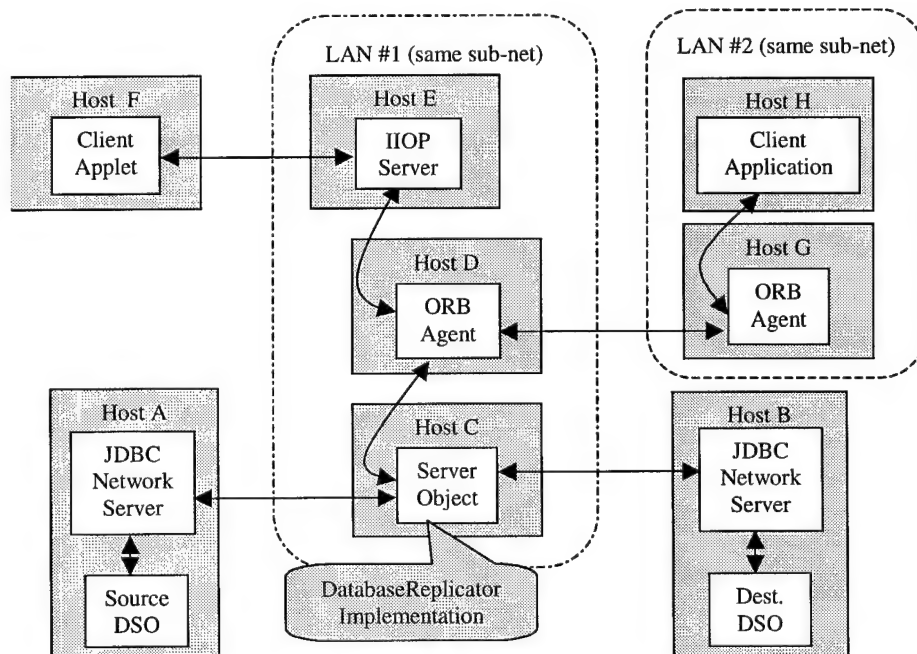
Figure 43. Distribution of processes for the three-tier/CORBA replication system.

86

The two components in Figure 43 not previously discussed are the *ORB agent* and the *IIOP service* (see Section 2.3.2.3). The ORB agent (Visigenic refers to their product as the ORB Smart Agent [Visigenic97]) provides the directory services necessary for the client and object implementation to locate each other. A client object uses broadcast messages over the network to locate and communicate with an ORB Agent. Broadcast messages are not considered appropriate for large networks, so the IIOP service (called the IIOP Gatekeeper by Visigenic) provides the support for clients outside the local sub-net to reach clients within the sub-net. The IIOP service acts as a proxy for clients outside the local sub-net, relaying their requests to locate an object to an ORB agent within the sub-net. The services of an ORB agent are sufficient to support client applications within a sub-net, but client applets operating outside the sub-net require the IIOP service. (Note that Visigenic's implementation of the IIOP service provides additional features that requires its use even if the client applet is executed within the sub-net.)

As already mentioned, the individual processes that must be operating to accomplish the three-tier replication approach can be distributed among several hosts. Of the many configurations possible for distributing the processes, the two of primary interest in this study are the one-host configuration (Figure 44) and four-host configuration (Figure 45).
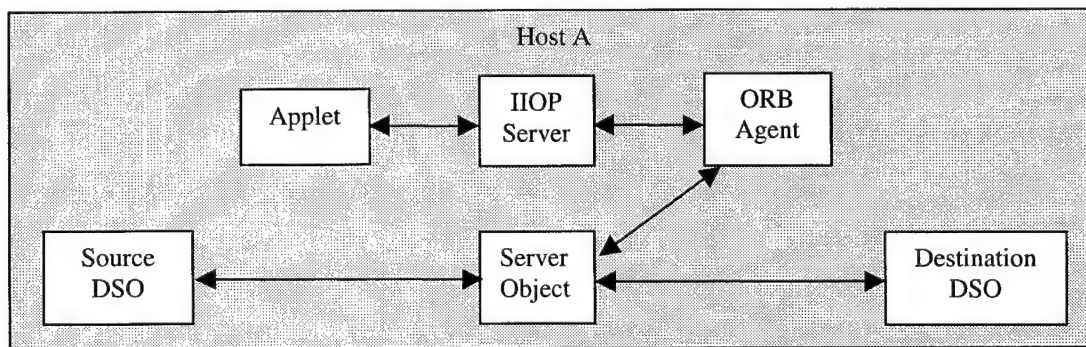
Figure 44. One-host configuration for three-tier/CORBA design.

The one-host configuration places all the processes on a single computer. This configuration supported development and initial testing of the three-tier implementation. Also, the one-host configuration offers the only completely controlled environment in which to make a

comparison with the two-tier approach. Note that the one-host configuration does not require JDBC network servers because the ODBC DSOs are located on the same host.
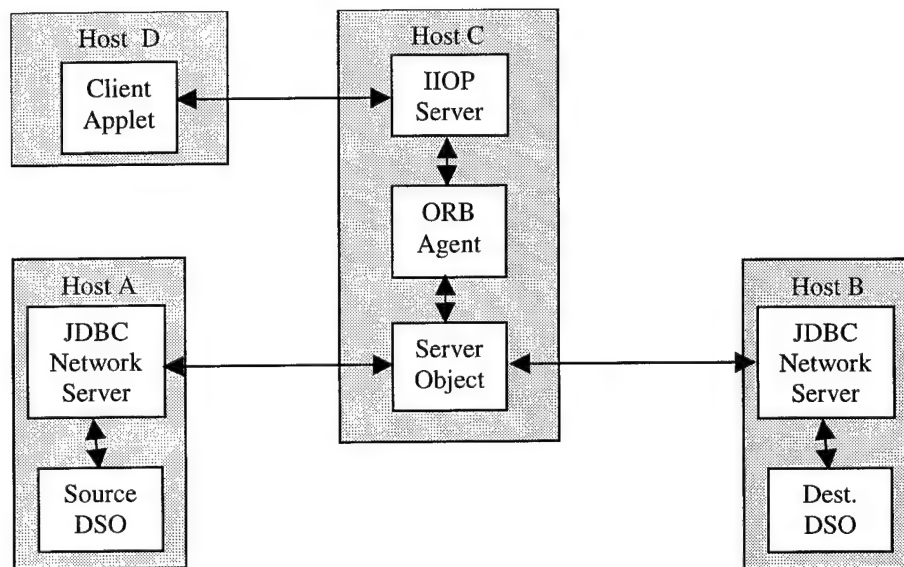
Figure 45. Four-host configuration for three-tier/CORBA design.

4.4 Test Preparation for the Two-tier vs. Three-tier Designs

To compare the relative performance of the two designs, several test cases were developed. Specifics of the test cases and rationale behind them are described below.

Each test case is broken down into *design, configuration, category*, and *data set. Design* identifies whether the test is two-tier or three-tier. *Configuration* identifies the number of hosts, specific client host platform (if applicable), and client network connection (if applicable) upon which the system is executed. *Test category* refers to whether the test case will be a homogeneous set of data (all columns in all records of all tables contain the same type and value) or a pre-defined data set. *Data set* indicates the specific data types and values used for the test— i.e., the data type for the homogeneous case or volume of data for the pre-defined data set. More detailed explanations of the test cases follow.

4.4.1 Configurations

The configurations used for testing vary by design approach, number of hosts, client applet host computer, and network connection between the client and the server processes. Table 3 lists the options available for each variable. Note that testing all combinations of these variables was not possible (given the test facilities) nor of value to this study. For example, the one-host configuration was limited to testing on a single platform—the development Windows NT machine–because that was the only system available capable of running the SQL Server RDBMS. Similarly, there were no options to change the network communication channel for the one-host configuration.

Table 3. Test configuration variables and options.

Configuration Variable	Options
Design Approach	Two-tier vs. Three-tier
Number of Hosts	1 and 3 for two-tier 1 and 4 for three-tier
Client Host	Sparc5 vs. 486-33
Network Connection	10Mbps Ethernet vs. 14.4Kbps dial-up

Meanwhile, the multiple host configurations—three-host for the two-tier and four-host for the three-tier—were chosen so as to put each tier of the entire replication system on a separate host computer, thus offering the most realistic "real-world" configuration. The actual system configurations used for testing are depicted in Table 4. For details of the hardware and software system specifications of the host computers, see Appendix C.

Table 4. Configurations for replication timing tests.

Number of Tiers	Number of Hosts	Client Platform	Communication Channel
2	1	Pentium-133	Local system
	3	Sparc5	Ethernet
		486-33	Ethernet
		486-33	14.4K modem
3	1	Pentium-133	Local system
	4	Sparc5	Ethernet
		486-33	Ethernet
		486-33	14.4K modem

Control of the configuration was partially accomplished by specifying the appropriate parameters in the HTML file in which each client applet was referenced. Through the HTML file, the specific applet could be chosen (i.e., two- or three-tier design) as well as the location of each DSO (via the appropriate URL). Appendix F contains examples of the HTML files for the one-host test configurations. Meanwhile, the selection of the communication channel was controlled by operating the 486-33MHz host either with the Ethernet connection or dial-up connection.

4.4.2 Test Categories and Data Sets

Two test case categories, *homogeneous* and *pre-defined*, were used for the comparison between the different designs and host configurations. Each homogeneous test case uses a source database that is constructed with tables containing columns of the same specified data type. Each such source database is built by the testing process immediately prior to the execution of the test. Each table within the source database is then populated with the specified number of records of a known value of that data type.

The *pre-defined* test case uses a pre-constructed and pre-populated database representative of the schema and content of CDSAR. An analysis of the data used by ACSC was used to define the minimal set of tables which are needed to support ACSC requirements. (See Appendix B for a brief discussion of this analysis.) A database was then constructed with this minimal set of tables and populated with data representative of that contained within the CDSAR database. The test cases are depicted as a tree in Figure 46, with each case being represented by the path from the root to a leaf node.

Specifically, the homogeneous tests all use a database having ten tables containing ten integer fields. Individual tests for the homogeneous test category populated the source database with 0, 1, 10, 100, and 1000 records, respectively. At four bytes per integer, the corresponding

volume of data[28] represented by these database sizes is approximately 0 kilobytes, 0.4 kilobytes, 3.9 kilobytes, 39 kilobytes, and 390 kilobytes, respectively.

Meanwhile, the pre-defined test case uses a pre-built database of the seven tables used by ACSC, with each table populated with approximately ten percent of the number of rows expected to be retrieved if this system were operated to support ACSC's needs. Based on the data type storage requirements for the Access database, the 10% volume of data equates to approximately 385 kilobytes.
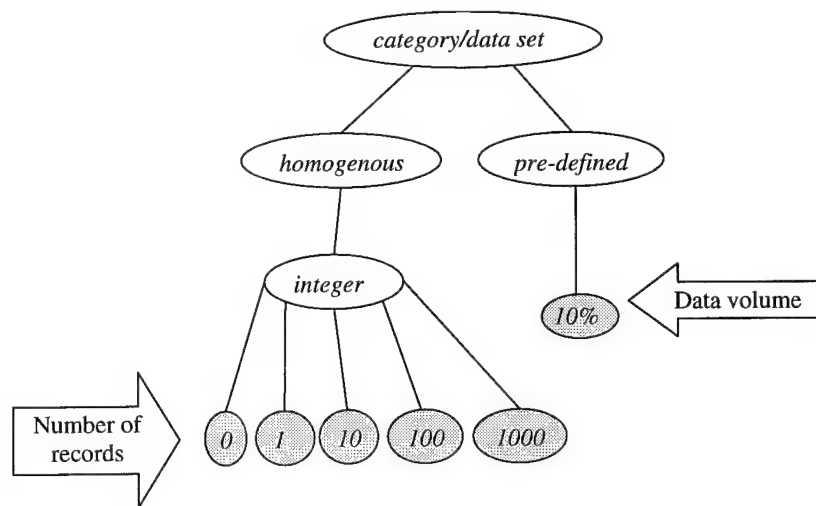
Figure 46. Test categories and data sets.

- Testing Limitations

The computers and networks upon which the multiple-host tests were executed are shared by the many students and faculty of AFIT. As a result, it was impossible to control the network traffic load and the CPU load on these common-use multi-tasking machines. Therefore, the multiple-host tests were conducted after duty hours (when resource usage is lower) to minimize the effects of other users on the systems.

---

[28] This refers to data only and not any additional storage overhead imposed by the DBMS—e.g., to support indexes, views, stored queries, etc.

Chapter 5 presents the results of conducting the replication timing tests outlined above. The results of those tests are used as a factor in selecting the design approach for the proof-of-concept system, also detailed in Chapter 5.

V. Results and Their Application to Proof-of-Concept System

This chapter presents the results of executing the tests outlined at the end of Chapter 4. The results of the testing indicate little performance difference between the two-designs under the single-host configuration. However, for the multi-host tests the three-tier design maintained consistent performance across different configurations while the two-tier design varied considerably. Based on these results, the three-tier design is chosen as the basis for the proof-of-concept system described later in this chapter. All data generated by the replication tests and used for the charts below is listed in Appendix D.

5.1 One-host Testing

The initial timing tests compared the replication time for a homogeneous set of data using the two-tier and three-tier designs. Figure 47 charts the resulting replication times and indicates that the performance of the two designs is similar for a given number of records. For the largest data volume (ten tables with ten fields and 1000 records per table), the replication times were within three percent of each other. These results suggest that the burden on the system from executing the three-tier design (which includes the additional CORBA support processes) is not significantly different from that of executing the two-tier design.
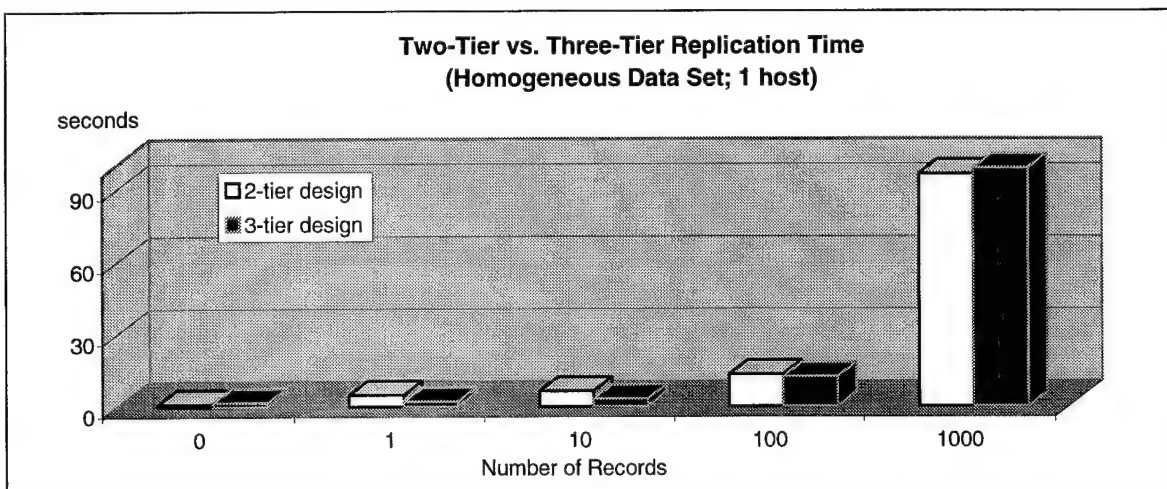
Figure 47. Comparison of replication times with all processes on one host.

5.2 Two-tier, Homogeneous Data Replication Times

Replication times for the two-tier design under the three-host configuration are charted in Figure 48. Note that replication times have now increased to several minutes, with the worst case—using the slower client platform with the low speed connection—requiring over 20 minutes for the 100-record case. Also, replication times became so great that the 1000-record test case was abandoned. (The first iteration of the Sparc5 client test for 1000 records required over 13 minutes to complete. The tests on the 486-33 client could have conceivably taken several hours each, and availability of hardware resources and reliability of the communications channel were not predictable enough for extended lengths of time.)

For the 100 record case, replication time for the 486-33 client with an Ethernet connection and the 486-33 client with a dial-up modem connection took 120% and 670% longer, respectively, when compared to the base case of replicating on the Sparc5. These results indicate that performance of the two-tier design is greatly affected by both the client's host capabilities and the client's communication channel.
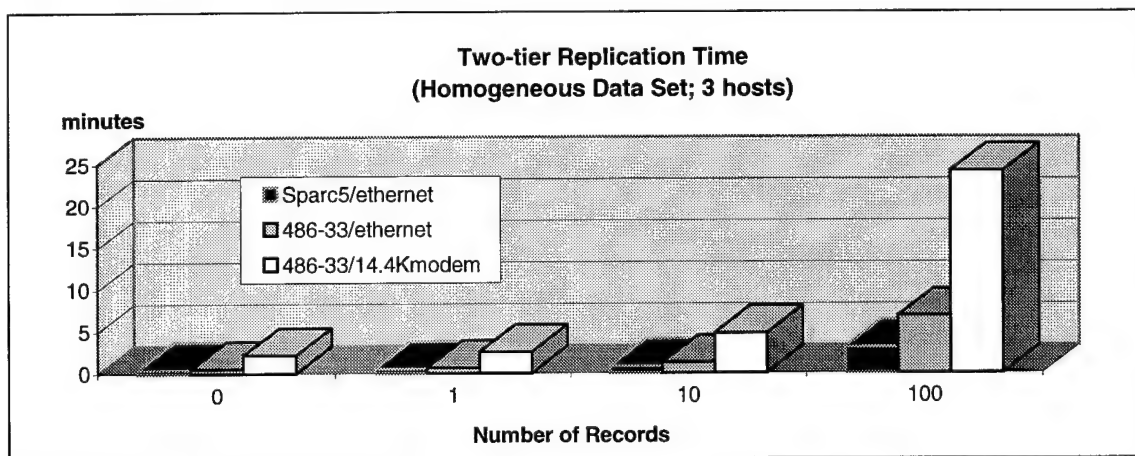
Figure 48. Comparing client platforms and client connections for two-tier.

5.3 Three-tier, Homogeneous Data Replication Times

Replication times for the three-tier test with homogeneous data did not vary significantly between platforms and communications channels. Figure 49 charts the times for replication.

(Note that the y-scale is in seconds, not minutes as with the previous chart.) This fairly consistent level of performance was expected due to the replication processing always occurring in the distributed object implementation on the same host. Surprisingly, the longest replication time occurred with the fastest client. One possible explanation for this slower performance is that the network traffic was unexpectedly heavy during the Sparc5 client test.

For the 100 record case, replication time for the Sparc5 client and the 486-33 client with a dial-up modem connection took 11% and 3% longer, respectively when compared to the base case of replicating on the 486-33 with the Ethernet connection. These results indicate that performance of the three-tier design is minimally affected by either the client's host capabilities or the client's communication channel.
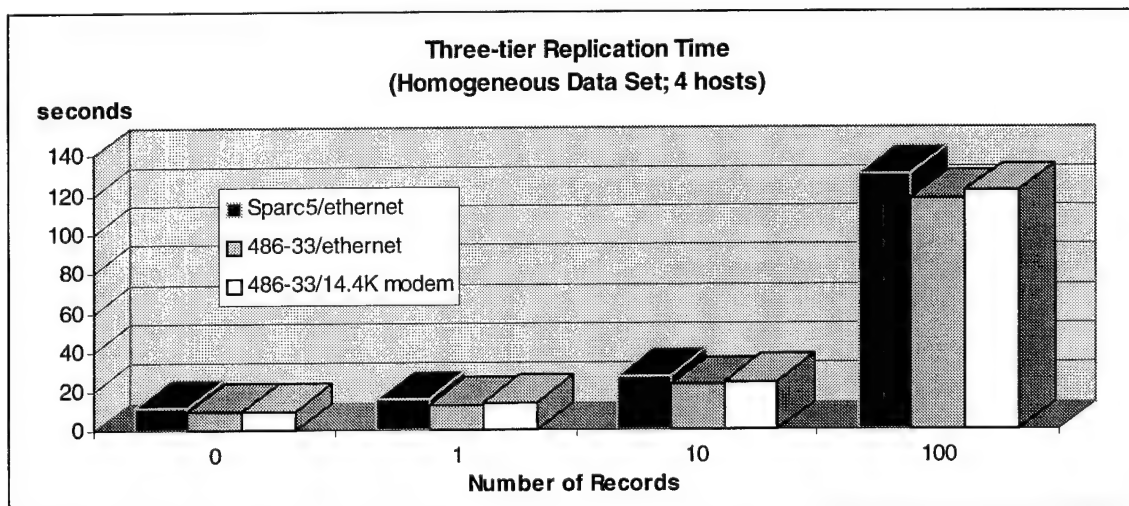
Figure 49. Comparing client platforms and client connections for three-tier.

5.4 Comparing Two-tier and Three-tier on Multiple Hosts

A side-by-side comparison of the replication times for the two multi-host designs using 100 records of the homogeneous data set is charted in Figure 50. Note that the two-tier design replication time increases as the client is changed to a slower machine and as the network connection is changed to a slower channel while the three-tier design's replication time stays

consistent. In the worst case, the two-tier design requires twelve times the total replication time

of the three-tier design.
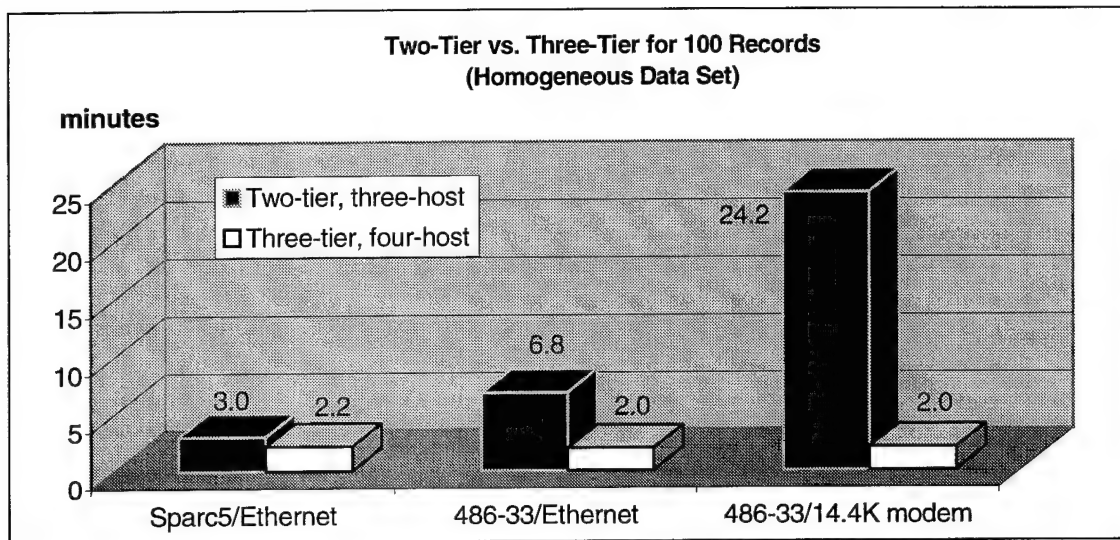
Figure 50. Comparing client platforms and client connections with both designs.

Recall that under the multi-host implementations of the designs in this study that

replication processing occurs on the client host for the two-tier system (see Figure 36 in Section

4.2) while replication occurs on the distributed object host for the three-tier system (see Figure 45

in Section 4.3.3). Consequently, by employing the three-tier architecture and making the client

largely a user interface to the application, client-side processing is greatly reduced over the two-

tier architecture.

Of course, the performance of the platform upon which the distributed server object is

executed becomes a determining factor for three-tier performance. In this study, the distributed

server object (i.e., the replication process) was executed on a relatively high performance host

computer (110Mhz Sparc5), thus producing both shorter and more consistent replication times.

The other factor in producing the more consistent and higher performance was the

communication channel between the host for the distributed server object and the hosts for the

DSOs. In this study, the server object's host computer was located on the same Ethernet network

as the computers hosting the two DSOs, thus providing a higher bandwidth connection to the databases.

5.5 Replicating 10% CDSAR Data

Replication of the pseudo-CDSAR data (see Figure 51) produced results similar to those found with the homogeneous data set tests. While the three-tier design required approximately 15 minutes in all cases, the two-tier design ranged from nineteen minutes to almost three hours.
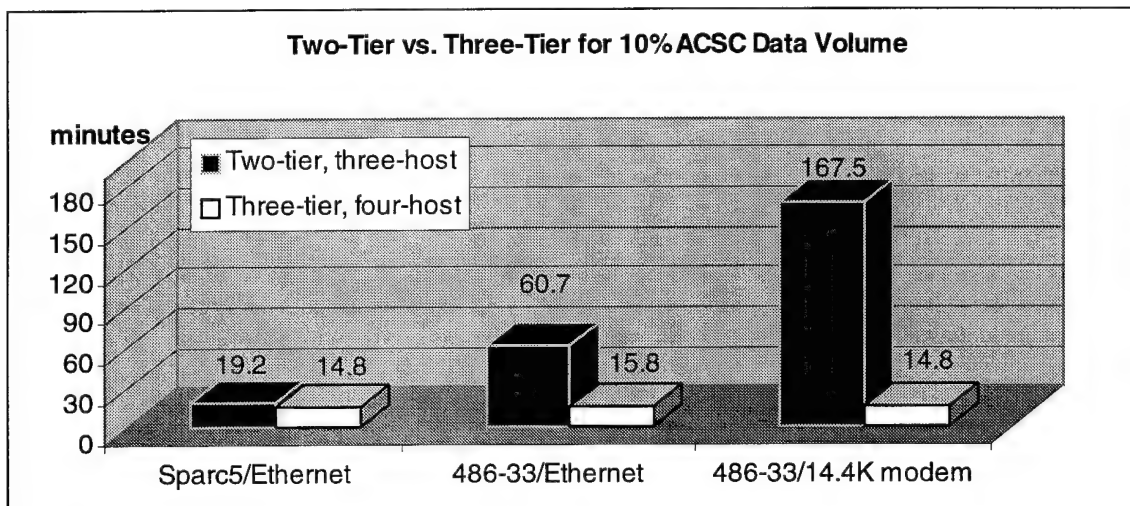
Figure 51. Comparing designs using 10% ACSC data.

5.6 Downloading Support Classes

Another consideration when comparing the use of the two- and three-tier designs is the overhead of downloading any additional Java class files needed to support the applet but that are not available on the client. For the two-tier design, the additional class files are the JDBC class files comprising the network-capable JDBC driver. For the three-tier design, the additional class files are the CORBA class files needed to support communication with the ORB and remote object. Table 5 shows the number of additional class files that were download to support operation of the replication test applets. The numbers represent those files that the client requests after the user presses the *Start* button within the applet to begin a test. (The class files initially

downloaded when the applet is first requested as part of the HTML page and those additional developer-produced class files requested after pressing the *Start* button are not included.)

If all of the necessary class files are available on the client, then no additional delay should be experienced. This proved true when testing the two designs on the development computer, as no measurable delay was experienced. However, when executing the two applets on client computers that did not have the additional files available locally, significant delay was introduced. Figure 52 presents the download times for the supporting class files for both the 10Mbps Ethernet and 14.4Kbps dial-up connection scenarios. Note that the worst case delay is for the 150 Visigenic CORBA library files that require over six minutes to download over the dial-up connection.

Table 5. Files downloaded to support applet operation.

Design	Support Provided	Number of Files	Total Size (KB)
Two-tier	Network-capable JDBC Driver	56	136
Three-tier	CORBA ORB and remote object access	150	593
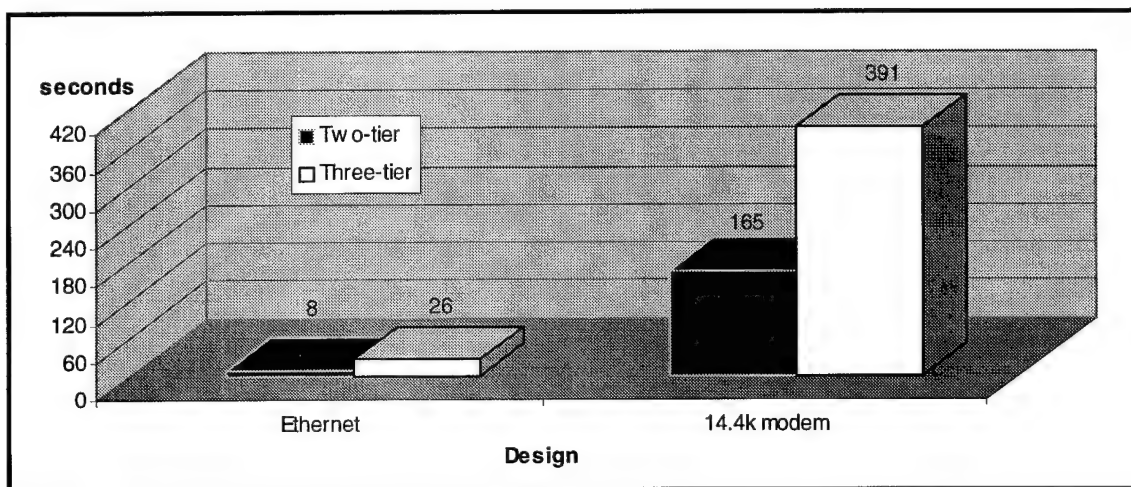
Figure 52. Download times for support class files.

5.7 Developing the Proof-of-Concept Version

The proof-of-concept system that follows is an attempt to demonstrate the potential benefits from using the technologies discussed so far. Having shown that the Java and CORBA

technologies are capable of supporting the mirrored-CDSAR approach discussed in Section 3.3 (although the performance may still not be satisfactory), this section looks at which design approach appears to provide the greater benefit and what can be done for the distance learning provider once the data has been replicated to a local DBMS.

5.7.1 Features

As presented above, the most consistent replication time was achieved with the three-tier design. Also, the infrastructure that operation of the CORBA ORB requires (as part of the three-tier design) could well support additional distributed objects whose services would also have more predictable performance. Therefore, this design was chosen as the basis for the replication portion of the proof-of-concept system. In addition, modifications were incorporated into the final system to enhance its usability and provide more automation for the distance learning provider.

The following features were chosen to integrate into the proof-of-concept system to demonstrate the potential of such a system:

1. Both automated and remote manual execution of the replication operation.

2. Filtering of replicated data to reduce the volume transferred.

3. Automated logging of replication activity.

4. Automated integration of appropriate replicated data into local distance learning provider database.

5. Automated staff notification of changes of interest in the database.

5.7.1.1 Automated and Manual Execution

The approach used to support both automated and remote manual operation was to create one client class that could be executed as either an applet or an application. For remote manual operation, the class can be referenced within an HTML page, thus allowing a user to manually activate the replication process from any Java-enabled web browser. Meanwhile, as an application the class can be executed as a "command line" program through a Java virtual

machine[29]. Because the application can be operated from a command line, it can be activated by any batch process or scheduling program. For example, the application could be executed as a batch file by the "at" scheduling program supplied with Windows NT. And due to the use of the three-tier/CORBA design, consistent performance could be expected no matter how the system was executed.

5.7.1.2 Reducing Data Transfer

To reduce the volume of data being transferred, the *DatabaseReplicator* class was modified to allow the incorporation of SQL WHERE clauses in the selection of data from the source DSO. The WHERE clauses are user-specified in a local text file, and each clause is identified with the table to which it applies. During the replication process, the *DatabaseReplicator* appends the appropriate WHERE clause to the SELECT statement used to query each source DSO table. Consequently, the distance learning provider staff can filter out unnecessary data from being transferred.

5.7.1.3 Logging Activity

The *DatabaseReplicator* class was also modified to log the replication activity of the *DatabaseReplicator* implementation object to a file on the server. Replication activity is written to a buffer during replication, then automatically saved to the specified disk file upon completion.

5.7.1.4 Data Integration

In preparation for integrating the data of interest within the replicated data set into the local distance learning provider's database[30], a test database was created representing a subset of the information a web-based distance learning provider might record. This small-scale database was managed by the same DBMS that controlled the destination DSO under the replication scenario. The tables defined within the database represent a student, a student's home contact

---

[29] For command-line operation, the program reads the initialization parameters from the same HTML file used for web browser execution. Reusing the same HTML applet parameters eliminates the need to maintain two separate configuration files.

[30] Note that the ultimate goal here is to produce one of the several course management system support databases described by Lockledge, et al. (see Section 2.4.1).

information, a student's work contact information, and a student's computing resources. (See Appendix E for the SQL Server SQL scripts that define the database schema.)

Next, a stored procedure was created within the new database. The stored procedure consisted of an SQL statement that searched for all occurrences of student social security numbers within the replicated database that did not occur within the distance learning provider database. Upon identifying non-matching social security numbers, the stored procedure creates new records in the distance learning provider's database using the data from the replicated database. Figure 53 contains the SQL Server stored procedure that accomplishes the steps just described. The stored procedure uses a nested INSERT statement to retrieve the data and map it to the distance learning provider's schema.

To activate the stored procedure, both the client and the *DatabaseReplicator* classes were modified. The IDL interface for *DatabaseReplicator* was modified by adding a method through which the database's stored procedure could be executed. The method *executeStoredProcedure* receives a string containing the name of the stored procedure, and the implementation of *DatabaseReplicator* uses a JDBC call to the connected DSO to execute the stored procedure. A new CORBA exception was also defined to communicate failure in the execution of the stored procedure. The client was modified to make use of the new *DatabaseReplicator* interface features by requesting the appropriate stored procedure be executed upon completion of the replication.

```
CREATE PROCEDURE AddNewACSCStudents AS
INSERT INTO ACSC.dbo.Student (SSAN,Rank,FirstName,MiddleName,LastName)
  SELECT ssn,rank_grade,first_name,middle_initial,last_name
  FROM stumast
  WHERE ssn NOT IN
    (SELECT ssan
    FROM ACSC.dbo.Student)
```

Figure 53. Stored procedure that updates distance learning provider database.

5.7.1.5 Staff Notification

Several approaches were possible to demonstrate the automated staff notification. Using SQL Server (or other sophisticated RDBMSs), it is possible to create a database *trigger*[31] that will generate an e-mail message containing information about the database—e.g., in this case that the database had changed because new students had been added. Using the "trigger and e-mail" technique would take advantage of the existing capabilities inherent in the DBMS. However, the resources available to the test platform upon which SQL Server was operating would not support the e-mail notification feature. (Specifically, a Windows NT mail server also had to be operating.)

Subsequently, as a partial step to demonstrate the notification process, a trigger was created that reacts to the insertion of records into the distance learning provider database by also inserting records into a separate table. The separate table represents a repository of only the new students in the system that require "in-processing" by the staff. The staff could be obligated by policy to review that table (and clear it after processing the new students). Or an additional feature could be added to the replication system that extracts the student data from the table of new students, builds an e-mail message, and sends it to the appropriate staff.

5.7.2 Results

Except for the e-mail notification feature already discussed, the proof-of-concept system functioned as expected. Using the features of the RDBMS—such as the stored procedures and triggers—proved relatively straight-forward. And although processing time was not recorded during the experiments with integrating the data, the perceived time for incorporating the new student data was negligible.

---

[31] Triggers are statements that are executed automatically by the DBMS in response to a change in the database. Depending on the sophistication of the DBMS, a trigger may be used to perform a wide range of actions, from enforcing integrity constraints to executing external programs. Not all RDBMSs support triggers—e.g., MS-Access does not support triggers while MS SQL Server does.

Incorporating the additional IDL for the new features and generating the new Java classes also was not difficult. While logging to file from the two-tier client would have raised security issues when operating within a web browser, this was not the case under the three-tier/CORBA design since the logging was directed by an object local to the system where the disk activity took place.

VI. Findings and Conclusions

This thesis examined the use of two new software technologies—Java and CORBA—to support access to legacy data used by Air Force distance learning providers. The results of this study demonstrate that these technologies are capable of enhancing existing systems that require relational data for their operation.

Chapter 2 provides terminology and background information in the areas covered by this thesis, with focus on the World Wide Web, Java, CORBA, and existing WWW-based course management systems. Chapter 3 provides an analysis of the operating environment of the ACSC distance learning system and presents a recommendation for improving that operation. Chapter 4 describes the design and implementation of two approaches—using two-tier and three-tier architectures—for the system described in Chapter 3. Chapter 5 describes the performance of the two designs of Chapter 4, with a subsequent recommendation to use the three-tier design for a proof-of-concept system.

6.1 Findings

This section begins with a general overview of the findings made during this research and follows with some specific issues surrounding use of the technologies that were applied.

6.1.1 Overview

The analysis of ACSC's operation revealed a strong, and growing, dependency on the CDSAR database. Any approach to enhancing web-based distance learning providers' capabilities must, therefore, address this dependency. As the analysis of Chapter 3 shows, ECI is a primary content provider (supplying student data) with CDSAR being the storage manager to whom distance learning consumers must turn. ACSC staff, as well as other Air Force web-based distance learning providers intending to expand, need an effective and efficient way to access CDSAR's data.

As discussed in Chapter 3, the CDSAR database does not incorporate several of the current standard relational features (e.g., primary and foreign keys), thus requiring that the system use external application logic to maintain the database. The reliance on external application logic and the limited remote access support (Telnet's command-line, character-based interaction) combine to restrict remote users to read-only access through an awkward interface. As such, innovative techniques must be used to provide the access that consumers of the CDSAR student data need for the future.

Providing access to this data through the mirrored-CDSAR approach (i.e., data replication) appears as one solution for better access. Data replication is an effective tool already employed in homogeneous distributed database environments. Replication offers the advantages of minimally affecting the local database and its operation while enabling (what appears to be) direct access for the users of that data—although in this case it is still read-only access.

Both the two-tier Java and three-tier Java/CORBA architectures support the heterogeneous, Internet-based replication approach. As the results of Chapter 5 indicate, however, the three-tier Java/CORBA system offered the benefit of providing consistent performance in various configurations of communications channel and platform. The performance at best, though, still required a replication time over a 10Mbps Ethernet connection of approximately 15 minutes for ten percent of the data volume expected. Assuming a linear extrapolation as the worst case scenario, this would equate to 150 minutes, or two and one-half hours, for the full data volume. Over a wide area network, such as the Internet connection between ACSC and ECI, the communications channel would likely be slower and the replication times longer than the best test case. This amount of time for replication would be unrealistic to support an operational site. Some reasons for the slow performance, as well as promises of improvements on the horizon, are discussed in Section 6.1.2, below.

However, if the replication system performance were to be improved, the benefits to be gained only begin with those demonstrated by the proof-of-concept applet described in Chapter 5. Having the CDSAR data as a local asset allows direct use of the features of the local RDBMS. And in the case of sophisticated RDBMSs this feature set includes automated database backup, transaction recovery, sophisticated report generation, stored procedures, triggers, advanced reports, and more (depending on the specific RDBMS product).

Recall from Section 3.1.3, that an ACSC staff member currently spends over 40 minutes per week to accomplish two student information updates to the web-based system. If the automated replication system discussed in this thesis (along with the associated operations supported by such a system) were incorporated into ACSC's operation, the time for the manual processing currently being done would likely be reduced to zero. Although the replication process is time-intensive, it could be executed as a batch process that occurs during periods of minimal system activity. Consequently, the data would be available for other automated utilities to process without staff intervention, and the staff could dedicate its time to other activities.

6.1.2 Issues Surrounding Technologies

The results of this study in applying Java and CORBA to the problem domain of Air Force distance learning providers reveals the viability of these technologies. However, the following issues should be kept in mind for development involving Java and CORBA.

6.1.2.1 Java Software Compatibility

Although billed as completely cross-platform, the Java language environment still suffers from compatibility problems. At this time Microsoft and Sun Microsystems are producing Java development environments and Java virtual machines that are not completely compatible with each other. Programs and applets created using one company's development environment may not work when executed by a Java virtual machine built by the other company (or according to the other company's specification).

This incompatibility problem appeared early in the research when products of Microsoft's development environment would not run within Sun's *Appletviewer* or Netscape's web browser. Also, the rapidly changing features of the Java specification (from Sun, which owns Java) meant that new versions of the Java development kit appeared almost monthly. Consequently, the decision was made early on in this research to use the Sun version of the Java development environment to take advantage of the latest Java features and have available the development tools, such as the *Appletviewer* utility, provided by Sun. Any developer choosing to use Java must be conscious of the possible incompatibilities that may appear when deploying software built with Java.

6.1.2.2 Java Virtual Machine Resource Consumption

Running individual Java virtual machines (JVM) for each Java program will consume resources on a host that a native, binary program will not. In the case of the three-tier systems implemented in this research, the four Java programs—applet, IIOP server, ORB agent, and replication server—each required their own JVM environment. With an earlier release of the CORBA ORB, the combination of the Java processes, DBMS server, and the web browser over-taxed the development system and consistently "crashed" at least one essential process when large data sets were used under one-host testing. (This was not a problem with the later release of the CORBA ORB.) The high number of processes, apparently, reduced the system virtual memory below the acceptable level. When the JVM is incorporated into the operating systems (as IBM's OS/2 already does [OS/2 Warp97] and others currently plan), this may reduce the amount of resources required to execute the individual Java processes and should improve performance of Java programs, in general.

6.1.2.3 Java Application and Virtual Machine Performance

As discussed in Section 2.3.3, Java is an interpreted language that is executed through a virtual machine. The initial popularity of Java stemmed from its use in deploying simple applets

with web pages. As such, performance was not critical since these small programs did not generally perform computationally intensive actions. However, as more complex Java programs are developed, the performance of the language becomes an issue. Such is the case with the performance of the replication system in this thesis.

To demonstrate the performance improvements on the horizon and otherwise available through different commercial products, informal tests were conducted at the end of this study using two additional software packages. The first software package is an enhancement to the Java Development Kit only recently made available and currently in beta test, called the *Java Performance Pack for Win32* [JPP97]. This software enhancement provides a Just-In-Time (JIT) compiler (see Footnote 11, page 30) that is reported by the developers, Sun Microsystems, to improve Java application performance up to ten times. In preliminary tests for a one-host replication configuration that employed JetConnect JDBC servers for the DSO connections, use of the JIT compiler improved performance (reduced replication time) by 27% over the same configuration without the Java Performance Pack. (Table 6 contains a listing of the test results.) Note that as its title indicates, this software will only provide a performance improvement on Win32 (i.e., Windows95 and Windows NT) systems.

The second product is another software manufacturer's implementation of a JDBC server. This product, called *VisiChannel for JDBC*, by Visigenic, uses its own CORBA ORB and IIOP to implement the JDBC driver services [VisiChannel97]. Use of the Visichannel for JDBC server improved performance (reduced replication time) by 49% compared to a similar configuration using JetConnect (without the Java Performance Pack). By also running the Java Performance Pack for Win32, the Visichannel for JDBC configuration improved replication performance by 56% over the JetConnect configuration (without the Java Performance Pack). Note that the server portion of this product is not available for the Windows95 platform, so it could not be incorporated into the multiple-host testing previously described in this thesis.

Table 6. **Performance improvements with JIT compiler and alternative JDBC server.**

Configuration	Replication Time (sec)	Percent of Baseline Time
JetConnect without JIT compiler	86	100% (baseline)
JetConnect with JIT compiler	63	73%
VisiChannel for JDBC without JIT compiler	44	51%
VisiChannel for JDBC with JIT compiler	38	44%

As seen by these results, performance improvements for Java applications are forthcoming. Those developers concerned about the performance of their Java applications must seek products that offer higher performance and must keep abreast of late-breaking developments in Java virtual machines.

6.1.2.4 CORBA Benefits and Costs

Employing a CORBA infrastructure along with CORBA-compliant objects offers considerable benefits to a distributed computing environment. By supporting the three-tier model, CORBA distributed applications can take advantage of the processing power, security controls, and fault tolerance of the server environment. And since CORBA is language-neutral, services of other distributed CORBA objects—available now or in the future—can be utilized no matter which popular programming language is used to create them.

However, implementation of CORBA application systems and development of three-tier systems requires considerable planning and investment. Thompson [Thompson97] describes the advantages of three-tier architecture over two-tier as improved scalability, business and technological flexibility, lower *long-term* cost, and higher-quality systems. Thompson also describes the challenges for using three-tier architecture, including the use of CORBA. Some of those challenges are high *short-term* cost, limited availability of training, few experienced people, incompatible standards, few commercially available distributed objects, limited choice of tools, and lack of end-user tools. Consequently, the use of CORBA and the three-tier architecture requires a reasonable organizational commitment with well-defined goals.

6.1.2.5 *Incompatibilities Among RDBMSs*

Although ODBC and JDBC provide an excellent method to access the features of and data maintained by RDBMSs, the incompatibilities among RDBMS products is still an issue. While the common denominator of SQL92 support allows a baseline of expectation when developing for a heterogeneous system, the fact remains that data types differ among RDBMSs— as seen in Section 4.2—as well as the specific features available—as mentioned with triggers in Section 5.7.1.

6.1.2.6 *Use of Applets*

The use of applets for supporting a client/server architecture raises the issues of program size and availability of class libraries. The benefits of using applets are the sophisticated features they can support (since they are based on a full-featured programming language), the ease with which they can be deployed, and the security mechanisms that are in-place during execution and available during development.

However, as the sophistication of an applet increases, so does its size and number of required class files. As discussed in Section 5.6, transmitting all the necessary class files from the server to the client can require an excess (and, likely, unacceptable) amount of time. Since many of the class files comprising an applet are from commercial libraries, the download time can be reduced by having these libraries pre-installed on the client's host. Note that the Java core API classes are, necessarily, included with each Java-enabled web browser so that they may be accessed quickly from the local host's environment.

In the specific case of client requirements for CORBA support, ORB classes may in the future be pre-installed with the web browser [Netscape97] or come with the operating system [SGI97]. Similarly, it is possible to have the application-specific classes that an applet requires pre-installed on each client host. However, installing the class files reduces the advantage of easy

deployment because the software manufacturer must manage software distribution through other mechanisms—e.g., mailing out disks or employing automated software update systems.

Another approach to reducing applet download time is the use of already available file compression schemes. The mechanism endorsed by Sun and supplied with its Java Development Kit is the creation of a Java archive, or *jar*, file that combines and compresses an applets components—including class, text, audio, video, etc.—all into one file that can be transmitted from the server to the client in one transaction. Microsoft offers a comparable mechanism with its Java development environment called the cabinet, or *cab*, file. However, web browser support for these two approaches is not yet standardized, so the software developer may risk not being able to deploy to all users.

6.2 Recommendations

Based on the knowledge gained from the background research and from designing and developing with these new technologies, the following recommendations are offered:

- **Incorporate a Local RDBMS.** Distance learning providers (such as ACSC) should, at a minimum operate a local, full-featured RDBMS that integrates with the web server to support management of the web-based correspondence student population. This will allow more automated maintenance of the web site and support sophisticated reporting on both the student population and web usage. Many of the automation features envisioned in ACSC's future (see Appendix A) can be implemented (or at least more easily supported) with the capabilities provided by sophisticated RDBMSs.

- **Work with ECI.** Air Force distance learning providers and ECI should work together to provide more direct and automated access to the student data within CDSAR. The approaches discussed in Chapter 3 of this thesis provide several

111

alternatives, with the mirrored-CDSAR (replication strategy) offering the best solution if ECI does not wish to change its current methods of operation. Although not directly studied in this thesis, the replication approach could employ a commercial homogeneous distributed database system; such a system would require installing an RDBMS compatible with that used by ECI at the distance learning provider site so that the replication features of that RDBMS can be fully utilized.

- **Develop with Java.** If web-based distance learning providers intend to develop additional features in-house, then they should teach their web developers Java. With Java comes the benefit of a common programming environment for applets (for client-side support), servlets (for server-side support), and distributed objects (using CORBA or RMI). ACSC is already taking advantage of the Perl language to support the Common Gateway interface, but Perl only offers features comparable to those provided by Java servlets. Many of the automation features envisioned in ACSC's future (see Appendix A) can be implemented using the wide range of programming features available with Java—especially, when accompanied by the support of a fully featured RDBMS.

- **Use CORBA Distributed Objects.** Incorporate a Java CORBA ORB into the distance learning provider's operating environment. This will support development and use of three-tier, session-oriented applications for both ACSC staff and students. Practically speaking, though, the application of CORBA would first require that a solution exist for the problem of long class library download times, as discussed above.

6.3 Conclusions

While the research presented in this thesis demonstrates the potential for the technologies of Java and CORBA, the specific software products from this study are not ready for real-world applications. With the lengthy replication times, this system is only feasible for small data volumes. However, the scalability and ease of use for systems using these technologies is quite high. For example, using the mirrored-CDSAR (replication) strategy, many AF distance learning providers could be added with each maintaining significant autonomy.

As the list of features desired by ACSC for its web site indicates, there is a wide range of capabilities that distance learning providers believe they should be able to provide. This thesis has presented only a small subset of the capabilities of several of the technologies that are available to support future distance learning. Other capabilities of Java and CORBA remain to be explored, as well as the other technologies both briefly mentioned (e.g., servlets, DCOM, and RMI) and not mentioned in this document.

6.4 Future Work

Many opportunities exist for future work based on the results of this thesis. Replication time for the small system developed in this study is not satisfactory for actual use. Therefore, effort could be directed at either improving the design of the software components involved or examining more efficient (higher performance) commercial components—e.g., faster JDBC network servers. Also, the "newness" of the specific implementations of Java, JDBC, and CORBA leaves the potential for rapid improvements in features and performance from the many manufacturers developing these core Java technologies. As the preliminary results discussed above suggest, an investigation of the newest releases of the Java virtual machines, CORBA ORBs, and JDBC implementations will likely reveal significance improvements in performance.

Different approaches to the replication process could also be explored—for example, extracting the data from the source into a data file, compressing the file, and transmitting the file

to the destination for incorporation into the local database. This would take advantage of the highly efficient bulk data operations many RDBMSs offer. Also, investigating the other approaches for CDSAR access, as discussed in Chapter 3, may reveal a more feasible technique for providing distance learning providers access to CDSAR data.

Security issues arise with the transferring of sensitive data, such as student Social Security account numbers, across clear channels. This issue is not addressed in this thesis. Similarly, the security issues surrounding access to and protection of the data at each site becomes more important as more mechanisms are incorporated to provide access.

If the approach in this thesis was to be expanded, an actual production database must be developed for ACSC. Effort to identify the exact needs of ACSC should be taken, with a focus on keeping any schema design flexible enough to serve other distance learning providers. Likewise, additional functional components supporting student and course administration should be explored, again with an emphasis on serving more than a specific distance learning provider.

This thesis emphasizes the use of RDBMSs. However, object databases and object-relational databases offer many of the features discussed above while also providing persistent storage and management for many of the other complex objects that web-based distance learning providers do and will use. Meanwhile, the trend in the development of software systems is to use an object-oriented approach for both design and implementation, a trend evidenced by the growing popularity of object-oriented languages like C++ and Java.

The role of object databases is to transparently support storage and management of the complex objects that comprise and are used by object-oriented software systems. As such, the integration of an object database to support the application objects, student information, and course content of distance learning might prove to be a better approach than attempting to manage and manipulate the disparate forms of information (such as relational student data vs. web page text and graphics) using separate, specialized storage systems. An evaluation of the

capabilities of sophisticated object databases and their applicability to the problem domain of Air Force web-based distance learning providers deserves study.

Finally, from an Air Force education and training perspective, a high-level study of the current distance learning programs offered by the Air Force is in order. Such a study must take into account (1) the role of centralized repositories (such as CDSAR) used by these programs, (2) the current and future requirements and goals for each program, and (3) the technologies used by the programs to accomplish their missions. Then, considering the new technologies available to support distance learning, both a model for future Air Force distance learning and a strategy for implementing such a model should be developed.

Bibliography

[2025 Report96]   "2025 Final Report Index." WWWeb,
                         http://www.au.af.mil/au/2025/report.htm (21 Oct 96).

[ACSC WWW97]   "The Air Command and Staff College Home Page." WWWeb,
                         http://wwwacsc.au.af.mil (16Apr97).

[Applet Security96]   "Frequently Asked Questions - Applet Security." WWWeb,
                         http://www.javasoft.com:81/sfaq/ (17 Dec 96).

[Aberdeen95]   "Universal Servers: RDBMS Technology for the Next Decade." *Informix
                         Technology Viewpoint*, Volume 9, Number 13. (June 3, 1995).
                         WWWeb,
                         http://www.informix.com/informix/corpinfo/zines/whitpprs/aberdeen/
                         aberdeen.htm#E (4 Nov 96).

[Atwood96]   Atwood, Thomas. "Object Databases Come of Age." *Object Magazine*
                         (July 1996): 60-63, 93.

[Baker96]   Baker, Brian. "Doing Business with the Web: The Informix/Illustra
                         Approach." *Advances in Database Technology - EDBT `96. Proceedings
                         from the 5th International Conference on Extending Database
                         Technology* (1996): 364-9.

[Bloor96]   "Oracle Universal Server from Oracle Corporation." WWWeb,
                         http://www.oracle.com/products/oracle7/Oracle_Universal_Server/
                         press/html/bloor_report.html (27 Nov 97).

[Booch91]   Booch, Grady. *Object Oriented Design with Applications*. Redwood
                         City: Benjamin/Cummings, 1991.

[Boutell96]   Boutell, Thomas. "World Wide Web Frequently Asked Questions"
                         WWWeb, http://www.boutell.com/faq/ (19 Sep 96).

[Bukhres96]   Bukhres, Omran A., Elmagarmid, Ahmed K. *Object-Oriented
                         Multidatabase Systems: A Solution for Advanced Applications*.
                         Englewood Cliffs: Prentice-Hall, 1996.

[Centra97]   "Centra Introduces The First Web-based Live Virtual Classroom for
                         Online Corporate Training Delivery." WWWeb,
                         http://www.centra.com/corporate/press/intropr.html (11 Apr 97).

[Chappell97]   Chappell, David, Linthicum, David S. "ActiveX Demystified." *Byte*
                         (September, 1997): 56-64.

[Cookies96]   "Persistent Client State HTTP Cookies." WWWeb,
                         http://www.netscape.com/newsref/std/cookie_spec.html (23 Nov 97).

[Criswick96]   Criswick, John R. "Digital Java Clocks." CD-ROM. *Java 1.1 Unleashed*. Second Edition. Version 1.0. Indianapolis: Sams Publishing, 1997.

[Date95]   Date, C. J. *An Introduction to Database Systems*, Sixth Edition. Reading: ACM Press, 1995.

[DB2-96]   "DB2 Universal Server." WWWeb, http://www.software.ibm.com/data/info/uni-server/db2unidb.html (14 Dec 96).

[Duan96]   Duan, Nick N. "Distributed database access in a corporate environment using Java." 1996. *Computer Networks and ISDN Systems,* vol. 28, no. 7-11 (1996): 1149-1156.

[Fritzinger96]   Fritzinger, J. Steven, Mueller, Marianne. "Java Security." WWWeb, http://www.javasoft.com/doc (15 Feb 97).

[Gosling95]   Gosling, James, McGilton, Henry. "The Java(TM) Language Environment: A White Paper." May 1996. WWWeb, http://www.javasoft.com/doc/language_environment/ (23 Feb 97).

[Gross97]   Gross, Christian. "Taking the Splash Diving into ISAPI Programming." *Microsoft Interactive Developer*. January 1997. WWWeb, http://www.microsoft.com/mind/0197/isapi.htm (22 Feb 97).

[Hamilton96a]   Hamilton, Marc A. "Java and the Shift to Net-Centric Computing." *IEEE Computer* (August 1996): 31-39.

[Hamilton96b]   Hamilton, Marc A., Cattell, Rick "JDBC: a Java SQL API." October 22, 1996. WWWeb, http://www.javasoft.com/nav/download/down.html (17 Feb 97).

[Harreld96]   Harreld, Heather. "Guidelines seen as core to Web policy." *Government Computer News* (December 2, 1996): 8,14.

[Hopson96]   Hopson K.C., Ingram, Stephen E. *Developing Professional Java Applets*. CD-ROM. *Java 1.1 Unleashed*. Second Edition. Version 1.0. Indianapolis: Sams Publishing, 1997.

[HTML96]   "HyperText Markup Language (HTML)." November 5, 1996. WWWeb, http://www.w3.org/pub/WWW/MarkUp/ (23 Oct 97).

[Hubler97]   Hubler, Alfred and Andrew Assad. "CyberProf: An Intelligent Human-Computer Interface for Asynchronous Wide-area Training and Teaching." WWWeb, http://cyber.ccsr.uiuc.edu/cyberprof/index.html (11 Apr 97).

[Iona97]   "OrbixWeb for Java." WWWeb, http://www.iona.com/Products/Orbix/OrbixWeb/index.html (3 Oct 97).

[Java Tutorial97]   "The Java Tutorial: Object-Oriented Programming for the Internet." July 8, 1997. WWWeb, http://java.sun.com/docs/books/tutorial/networking/sockets/ index.html (8 Oct 97).

[JavaOS96]   "JavaOS(TM): The Standalone Java(TM) Application Platform - White Paper." October 29, 1996. WWWeb, http://www.javasoft.com/products/javaos/index.html (15 Nov 96).

[Javascript96]   "JavaScript Guide." 1996. WWWeb, http://home.netscape.com/eng/mozilla/3.0/ handbook/javascript/ index.html (3 Feb 97).

[JDBC97]   "JDBC™ Drivers." September 24, 1997. WWWeb, http://www.javasoft.com/products/jdbc/jdbc.drivers.html (9 Oct 97).

[Joe97]   "Joe." WWWeb, http://www.sun.com/solaris/neo/joe/ (3 Oct 97).

[JPP97]   "README: Java Performance Pack for Win32." WWWeb, http://www.javasoft.com/products/jdk/1.1/ (5 Oct 97).

[Korth97]   Korth, Henry F., Silberschatz, Abraham. *Database System Concepts*. New York: McGraw-Hill, 1986.

[Kristol95]   Kristol, David M. "Proposed HTTP State-Info Mechanism." September 1995. WWWeb, http://204.178.16.7/~dmk/session.html (4 Dec 97).

[Larson95]   Larson, James A. *Database Directions*. Upper Saddle River: Prentice-Hall, 1995.

[LearningSpace97]   "New Education Solution – LearningSpace: Distributed, Collaborative Learning." WWWeb, http://www2.lotus.com/education.nsf (11Apr97).

[LearnLinc97]   "LearnLinc™ I-Net." WWWeb, http://www.ilinc.com/virtual.htm (11 Apr 97).

[Liederman96]   Liederman, Erica. "Sybase's New Web Tools." July-September 1996. *Sybase On Line*. WWWeb, http://www.sybase.com/inc/sybmag/quarter3_96/workplace/internet.html #tools (7 Nov97)

[Lockledge96]   Lockledge, Jeffrey, Geister, Donald, Mitchell, Lance. "An Internet Based Learning Environment." *IEEE Frontiers in Education '96 Proceedings (Volume 1)*: 293-296.

[Locklin97a]   Locklin, Charles. Telephone interview. 12 Jun 97.

[Locklin97b]   Locklin, Charles. "CDSAR Schema." Personal e-mail (11 Jun 97).

[Malkin96]   Malkin, Gary S. "RFC1983: Internet Users' Glossary." August 1996. WWWeb, http://globecom.net/(nobg)/ietf/rfc/rfc1983.shtml (9 Oct 97).

118

[Melton96]   Melton, Jim. "An SQL3 Snapshot." *Proceedings of the Twelfth International Conference on Data Engineering* (1996): 666-72.

[Miranda96]   Miranda, Jose Eduardo Pina, Pinto, Jorge Sousa. "Using Internet Technology for Course Support." *SIGCSE Bulletin (Special Edition) Integrating Technology in Computer Science Education*, vol. 28 (1996): 96-98.

[Montinola96]   Montinola, Katrina. "Oracle Web Server: A Technical Discussion." WWWeb, http://tiburon.us.oracle.com/odp/public/library/cr/html/cr_white.html (9 Dec 96).

[Morrison97]   Morrison, Michael, et. al. *Java Unleashed*, Second Edition. Indianapolis: Sams.net, 1997.

[NCSA-CGI]   "Common Gateway Interface." WWWeb, http://hoohoo.ncsa.uiuc.edu/cgi/ overview.html (21 Nov 97).

[Netscape97]   "DevEdge Online - Netscape ONE – CORBA." June 10, 1997. WWWeb, http://developer.netscape.com/one/components/corba/index.html (8 Oct 97).

[Nishida96]   Nishida, Tomohiro, Saitoh, Akinori, Tsujino, Yoshihiro, Tokura, Hobuki. "Lecture Supporting System by Using E-mail and WWW." *SIGCSE Bulletin*, vol. 28, no. 1 (1996): 280-282.

[Objectstore96]   "ObjectStore, The Natural Database Solution for the Web!" WWWeb, http://www.odi.com/products/onweb/overview.html (23 Nov 96).

[ODBC96]   "Understanding ODBC and OLE." WWWeb, http://www.microsoft.com/odbc/ wpapers/odbcnole.htm (23 Nov 96).

[ODMG93]   "The Object Database Standard: ODMG-93." WWWeb, http://www.odmg.org/odmg-93.html (21 Oct 97).

[OMG96]   "What is Corba?" WWWeb, http://www.omg.org/corba.htm (21 Oct 96).

[Oracle96]   "Oracle 7 Release 7.3: Products and Options." 1996 WWWeb, http://www.oracle.com/products/oracle7/oracle7.3/ (13 Dec 96).

[Orfali97]   Orfali, Robert, Harkey, Dan. *Client/Server Programming with Java and CORBA*. New York: John Wiley & Sons, 1997.

[OS/2 Warp97]   "OS/2 Warp 4 / The Premiere Networking Client" WWWeb, http://www.software.ibm.com/os/warp/warp-client/ (8 Oct 97).

[Rauch96]   Rauch, Wendy B. *Distributed Open Systems Engineering*. New York: John Wiley & Sons, 1996.

[Rumbaugh91]   Rumbaugh, James; Blaha, Michael; Premerlani, William; Eddy,
                 Frederick; Lorensen, William. *Object-Oriented Modeling and Design.*
                 Englewood Cliffs: Prentice Hall, 1991.

[SGI97]   "Silicon Graphics Selects Visigenics Leading IIOP/CORBA
                 Technology." August 15, 1997. WWWeb,
                 http://www.nasdaq.com/sitelayout.asp?section=/analytics/analytics.asp?
                 symbol=VSGN (8 Oct 97).

[SHTTP96]   "Secure WWWEB, HTTP." WWWeb, http://www.eit.com/creations/s-
                 WWWeb, http/ (13 Nov 96).

[Sikes96]   Sikes, Carol S., Cherry, Adelaide K., Durall, William E., Hargrove,
                 Michael R., Tingman, Kenneth R. "Brilliant Warrior: Information
                 Technology Integration in Education and Training." August 1996.
                 WWWeb, http://www.au.af.mil/au/2025/volume1/chap10/v1c10-1.htm
                 (21 Oct 96).

[Singh96]   Singh, T., Zhu, M. Thakkar, U., Ravailoi. "Impact of World Wide Web,
                 Java, and Virtual Environments on Education in Computational Science
                 and Engineering." *IEEE Frontiers in Education '96 Proceedings*, vol. 3
                 (1996): 1007-1010.

[SQL Server]   "Microsoft SQL Server in the Active Internet: Product Strategy
                 Backgrounder" WWWeb, http://www.microsoft.com/sql/sqlinet1.htm (3
                 Feb 97).

[SSL96]   "The SSL Protocol." 1996. WWWeb,
                 http://home.netscape.com/newsref/ std/SSL.html (22 Jan 97).

[Stein96]   Stein, Lincoln D. "The World Wide Web Security FAQ, Version 1.3.0."
                 November 8, 1996. WWWeb,
                 http://www.genome.wi.mit.edu/www/faqs/www-security-faq.html (10
                 Feb 97).

[Swafford96a]   Swafford, Michael L., Graham, Charles R., Brown, Donna J., Trick,
                 Timothy N. "Mallard™: Asynchronous Learning in Two Engineering
                 Courses." *IEEE Frontiers in Education '96 Proceedings*, vol. 3 (1996):
                 1023-1026.

[Swafford96b]   Swafford, Michael L., Brown, Donna J. "Mallard™: Asynchronous
                 Learning on the World-Wide Web." *Proceedings of the ASEE 96
                 Conference* (1996).

[Thompson97]   Thompson, Charles. "Three-tier Architecture." *Database Programming
                 and Design*, vol. 10, no. 8 (August 1997): 27-33.

[Tinoco97]   Tinoco, Lucio, Fox, Edward A., Barnette, N. Dwight. "On-line
                 Evaluation in WWW-based Courseware." *SIGCSE Bulletin*, vol. 29, no.
                 1 (March 1997): 194-198.

[TopClass97]   "Guided Learning: Using the TopClass™ Server as an Effective Web-
                    based Training System (A WBT Systems White Paper)." WWWeb,
                    http://www.wbtsystems.com/index.html (11May97).

[Vice97]   Vice, John. "fwd: re: More Questions." Personal e-mail (24Sep97).

[VisiChannel97]   "VisiChannel for JDBC." WWWeb
                    http://www.visigenic.com/prod/vc4jdbc/ (24 Oct 97).

[Visigenic97]   "Visigenic VisiBroker for Java." WWWeb,
                    http://www.visigenic.com/prod/vbrok/ vb30DS.html (3Oct97).

[Whetzel96]   Whetzel, John K. "Integrating the World Wide Web and Database
                    Technology." March/April 1996. *AT&T Technical Journal*. 38-46.

[WC3-96]   "W3C - The World Wide Web Consortium." December 9, 1996.
                    WWWeb, http://www.w3.org/pub/WWW/Consortium/ (4 Feb 97).

[WebCT97]   "World Wide Web – Course Tool: An Environment for Building WWW-
                    Based Course" WWWeb,
                    http://homebrew.cs.ubc.ca/webct/papers/p29/index.html (11 Apr 97).

Appendix A.   ACSC Proposed Course Management System Features

The course management system envisioned by ACSC [ACSC DL97] will have the capability to:

- Serve course materials to approximately 9,000 students.

- Communicate with students via the world wide web in a secure manner using encryption protocols.

- Perform a "hot" backup of the entire system, including all course materials and associated databases.

- Provide an on-line enrollment system with automated creation and distribution of accounts, passwords, and student welcome information.

- Provide appropriate access into the student and resource databases for all students, faculty, and administrative staff.

- Prepare and present reports describing the performance and usage of the system and progress of students.

- Prepare individual student lessons based on test results.

- Present student self-evaluations at designated times in a course.

- Monitor and record the location of each student within the course so as to support immediate return to the same location upon re-entry.

- Allow students to record notes associated with the on-line material, which may then be posted to an on-line bulletin board, printed, or e-mailed.

- Provide embedded links in all materials to support contact with administrative staff and course developers.

- Prepare and distribute automated course reminders to students–e.g., e-mail notifications of upcoming test dates, course changes, web page updates, etc.

- Perform complete on-line testing, to include test creation from a database of lesson-specific questions, test administration to individual students, immediate grading, immediate feedback, and automated recording of test scores.

- Analyze test results and prepare follow-on lessons as appropriate to address individual student's weak areas.

- Allow students to provide immediate feedback about each test.

- Maintain test results in a database and make the results database available for review and analysis.

- Provide a virtual classroom to simulate the in-person seminars associated with the ACSC course. The virtual classroom will allow tele-conferencing via text, voice, or video, and will provide a shared whiteboard and automated logging of seminar transcripts.

- Support a virtual, on-line war-gaming scenario.

- Maintain a virtual library as a repository for research papers and other scholarly works.

- Provide a forum for debate using either an electronic bulletin board or an e-mail automated list server.

- Support streaming multimedia storage and presentation–i.e.., streaming audio and video.

- Support a virtual auditorium where students can listen to or view faculty lectures, guest speakers, public debates, etc.

An analysis of the CDSAR database based on correspondence with ECI and ACSC was performed to identify which data was of interest to ACSC. Table 7 lists the number of objects that exists within the CDSAR database. ACSC maintains a set of 31 (at the time of this writing) SQL queries that are used to retrieve information from CDSAR. An analysis of the queries revealed that ACSC only made use of the data from seven of the 129 tables. The number of records for each of these tables as of 21August97, listed in Table 8, ranges from under 1000 to over 1 million.

Table 7. Number of database objects stored in CDSAR.

Object	Count
Table	129
Index	135
View	26
Procedure	5
Function	1
Trigger	1
Package	3
Synonym	51

Table 8. Number of records in the tables of interest to ACSC.

Table	Number of Records
anstat	87306
crse	684
enrlhist	1143165
enrlmast	158460
hsummary	148732
stumast	143820
summary	91115

Appendix C.   Host System and Software Specifications for Development and Testing

Table 9. Host system specifications.

Computer System	CPU	RAM	Operating System	Roles
Development Platform	Intel Pentium 133MHz	40MB	Microsoft Windows NT Server 4.0 (evaluation version)	Development host, Destination DSO host, Host for one-host test
486-33 Client	Intel 486 33MHz	8MB	Microsoft Windows95	Client and DSO host for initial two-host experiments, Client host for two- and three-tier tests
Windows95 Source DSO Host	Intel Pentium 120MHz	64MB	Microsoft Windows95	Source DSO host for two- and three-tier tests
Sparc5 Client	S5 85MHz	48MB	Solaris 2.5.1	Client host for two- and three-tier tests
Sparc5 Server	S5 110MHz	96MB	Solaris 2.5.1	Host for DatabaseReplicator server object, ORB agent, IIOP server, and web server during two- and three-tier tests

Table 10. Additional software used for development and testing.

Purpose	Product	Version
Web browser for timing tests	Sun HotJava Browser	1.1, beta 2 for Solaris, 1.1, beta 2 for Windows95/NT
Java Development Environment	Sun Java Development Kit	1.1.3 for Solaris, 1.1.3 for Windows95/NT
Source DBMS-multiple host timing tests	Microsoft Access	7.0 (Windows95/NT only)
Source DBMS-development and one host timing test	Microsoft Access	Access97/SR-1 (Windows95/NT only)
Destination DBMS	Microsoft SQL Server (evaluation version)	6.5 (Windows NT only)
CORBA Java ORB	Visigenic Visibroker for Java (evaluation version)	3.0 (initial research used versions 2.0 and 2.5)
JDBC Network Server Package	XDB Systems JetConnect	2.05

Appendix D.   Results Data from Replication Tests

<u>One-host Test Results</u>

The one-host replication testing was conducted using the Sun JDBC-ODBC bridge to connect to both DSOs. The results below reflect the average of 10 iterations of each test case.

Table 11. Two-tier, one-host test results data.

iterations	test	data type	tables	records	columns	msec	sec
10	type	integer	10	0	10	851	0.9
10	type	integer	10	1	10	4736	4.7
10	type	integer	10	10	10	6310	6.3
10	type	integer	10	100	10	13239	13.2
10	type	integer	10	1000	10	96178	96.2

Table 12. Three-tier, one-host test results data.

iterations	test	data type	tables	records	columns	msec	sec
10	type	integer	10	0	10	1522	1.5
10	type	integer	10	1	10	1983	2.0
10	type	integer	10	10	10	3084	3.1
10	type	integer	10	100	10	12228	12.2
10	type	integer	10	1000	10	98402	98.4

Two-tier Test Results

The tables below contain the raw test results from the two-tier tests involving three hosts and the homogeneous integer data set. All tests used the XDB Systems JDBC server and driver for DSO connectivity.

Table 13. Two-tier, Sparc5 client (Ethernet) test results data.

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)
1	type	integer	10	0	10	28225	
2	type	integer	10	0	10	28600	
3	type	integer	10	0	10	29081	
4	type	integer	10	0	10	29344	
5	type	integer	10	0	10	29332	28.9
1	type	integer	10	1	10	31521	
2	type	integer	10	1	10	31905	
3	type	integer	10	1	10	32012	
4	type	integer	10	1	10	33102	
5	type	integer	10	1	10	32524	32.2
1	type	integer	10	10	10	47041	
2	type	integer	10	10	10	47888	
3	type	integer	10	10	10	49140	
4	type	integer	10	10	10	49078	
5	type	integer	10	10	10	52041	49.0
1	type	integer	10	100	10	174627	
2	type	integer	10	100	10	182537	
3	type	integer	10	100	10	182603	
4	type	integer	10	100	10	186428	
5	type	integer	10	100	10	187913	182.8

Table 14. Two-tier, 486-33MHz client (Ethernet) test results data.

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)
1	type	integer	10	0	10	35040	
2	type	integer	10	0	10	33610	
3	type	integer	10	0	10	33610	
4	type	integer	10	0	10	33780	
5	type	integer	10	0	10	34990	34.206
1	type	integer	10	1	10	39600	
2	type	integer	10	1	10	39710	
3	type	integer	10	1	10	40870	
4	type	integer	10	1	10	40810	
5	type	integer	10	1	10	42890	40.776
1	type	integer	10	10	10	75850	
2	type	integer	10	10	10	76340	
3	type	integer	10	10	10	74530	
4	type	integer	10	10	10	78430	
5	type	integer	10	10	10	77830	76.596
1	type	integer	10	100	10	401890	
2	type	integer	10	100	10	413040	
3	type	integer	10	100	10	398650	
4	type	integer	10	100	10	413470	
5	type	integer	10	100	10	421330	409.676

Table 15. Two-tier, 486-33MHz Client (14.4Kbps dial-up) test results data.

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)
1	type	integer	10	0	10	129730	
2	type	integer	10	0	10	136540	
3	type	integer	10	0	10	130560	
4	type	integer	10	0	10	130450	
5	type	integer	10	0	10	131380	131.7
1	type	integer	10	1	10	148730	
2	type	integer	10	1	10	149020	
3	type	integer	10	1	10	152520	
4	type	integer	10	1	10	157420	
5	type	integer	10	1	10	155110	152.6
1	type	integer	10	10	10	277430	
2	type	integer	10	10	10	282260	
3	type	integer	10	10	10	289790	
4	type	integer	10	10	10	286050	
5	type	integer	10	10	10	296160	286.3
1	type	integer	10	100	10	1450480	1450.5

Three-tier Test Results

The tables below contain the raw test results from the three-tier tests involving four hosts and the homogeneous integer data set. All tests used the XDB Systems JDBC server and driver for DSO connectivity.

Table 16. Three-tier, Sparc5 client (Ethernet) test results data.

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)
1	type	integer	10	0	10	11753	
2	type	integer	10	0	10	12033	
3	type	integer	10	0	10	12215	
4	type	integer	10	0	10	12098	
5	type	integer	10	0	10	12676	12.155
1	type	integer	10	1	10	15688	
2	type	integer	10	1	10	16117	
3	type	integer	10	1	10	17193	
4	type	integer	10	1	10	16148	
5	type	integer	10	1	10	16728	16.3748
1	type	integer	10	10	10	25234	
2	type	integer	10	10	10	25874	
3	type	integer	10	10	10	27225	
4	type	integer	10	10	10	28898	
5	type	integer	10	10	10	29022	27.2506
1	type	integer	10	100	10	125396	
2	type	integer	10	100	10	126772	
3	type	integer	10	100	10	130360	
4	type	integer	10	100	10	133224	
5	type	integer	10	100	10	136806	130.5116

Table 17. Three-tier, 486-33MHz client (Ethernet) test results data.

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)
1	type	integer	10	0	10	9060	
2	type	integer	10	0	10	9330	
3	type	integer	10	0	10	9730	
4	type	integer	10	0	10	9610	
5	type	integer	10	0	10	9940	9.534
1	type	integer	10	1	10	11590	
2	type	integer	10	1	10	12080	
3	type	integer	10	1	10	12470	
4	type	integer	10	1	10	13890	
5	type	integer	10	1	10	13020	12.61
1	type	integer	10	10	10	21910	
2	type	integer	10	10	10	22790	
3	type	integer	10	10	10	22360	
4	type	integer	10	10	10	23620	
5	type	integer	10	10	10	23720	22.88
1	type	integer	10	100	10	112980	
2	type	integer	10	100	10	117650	
3	type	integer	10	100	10	117480	
4	type	integer	10	100	10	120230	
5	type	integer	10	100	10	120450	117.758

Table 18. Three-tier, 486-33MHz client (14.4Kbps dial-up) test results data.

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)
1	type	integer	10	0	10	9830	
2	type	integer	10	0	10	10050	
3	type	integer	10	0	10	10170	
4	type	integer	10	0	10	10600	
5	type	integer	10	0	10	10710	10.272
1	type	integer	10	1	10	13900	
2	type	integer	10	1	10	13620	
3	type	integer	10	1	10	13620	
4	type	integer	10	1	10	13180	
5	type	integer	10	1	10	15380	13.94
1	type	integer	10	10	10	24610	
2	type	integer	10	10	10	23900	
3	type	integer	10	10	10	25650	
4	type	integer	10	10	10	23670	
5	type	integer	10	10	10	25490	24.664
1	type	integer	10	100	10	116720	
2	type	integer	10	100	10	119680	
3	type	integer	10	100	10	119960	
4	type	integer	10	100	10	124240	
5	type	integer	10	100	10	125780	121.276

10% CDSAR Data Volume Test Results

The tables below contain the results from the two- and three-tier tests using 10% of the expected CDSAR data volume that ACSC would replicate under operational conditions. All tests used the XDB Systems JDBC server and driver for DSO connectivity. Only one iteration was executed for each case due to limitations on time and testing resources.

Table 19. Results data for 10% CDSAR data volume replication using two-tier design.

Two-tier, 486-33MHz, 14.4K dial-up:

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)	avg rep time(min)
1	non	int	1	1	1	10052030	10052.03	168

Two-tier, 486-33MHz, Ethernet:

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)	avg rep time(min)
1	non	int	1	1	1	3644530	3644.53	61

Two-tier, Sparc5, Ethernet:

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)	avg rep time(min)
1	non	int	1	1	1	1149348	1149.348	19

Table 20. Results data for 10% CDSAR data volume replication using three-tier design.

Three-tier, 486-33MHz, 14.4K dial-up:

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)	avg rep time(min)
1	non	int	1	1	1	888860	888.86	15

Three-tier, 486-33MHz, Ethernet:

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)	avg rep time(min)
1	non	int	1	1	1	946860	946.86	16

Three-tier, Sparc5, Ethernet:

Test No.	Test type	Data type	Tables	Records	Columns	rep time (msecs)	avg rep time(sec)	avg rep time(min)
1	non	int	1	1	1	888532	888.532	15

```
/****** Object:  Table dbo.NewStudent      Script Date: 8/27/97 4:26:53 PM
******/
CREATE TABLE "dbo"."NewStudent" (
  "SSN" varchar (9) NOT NULL ,
  "Rank" varchar (5) NULL ,
  "FirstName" varchar (15) NULL ,
  "MiddleInitial" varchar (1) NULL ,
  "LastName" varchar (15) NULL ,
  "Suffix" varchar (20) NULL ,
  "DateUpdated" "datetime" NULL ,
  "Notes" "text" NULL ,
  CONSTRAINT "PK_NewStudent_1__10" PRIMARY KEY  CLUSTERED
  (
    "SSN"
  )
)
GO

/****** Object:  Table dbo.Student     Script Date: 8/27/97 4:26:53 PM ******/
CREATE TABLE "dbo"."Student" (
  "SSN" varchar (9) NOT NULL ,
  "Rank" varchar (5) NULL ,
  "FirstName" varchar (15) NULL ,
  "MiddleInitial" varchar (1) NULL ,
  "LastName" varchar (15) NULL ,
  "Suffix" varchar (20) NULL ,
  "DateUpdated" "datetime" NULL ,
  "Notes" "text" NULL ,
  CONSTRAINT "PK_Student_1__10" PRIMARY KEY  CLUSTERED
  (
    "SSN"
  )
)
GO

/****** Object:  Table dbo.ComputerSystem     Script Date: 8/27/97 4:26:53 PM
******/
CREATE TABLE "dbo"."ComputerSystem" (
  "SSN" varchar (9) NOT NULL ,
  "CPU" varchar (10) NULL ,
  "ModemSpeed" varchar (50) NULL ,
  "CDROMSpeed" varchar (10) NULL ,
  "Memory" varchar (50) NULL ,
  "WebBrowser" varchar (20) NULL ,
  "HardDriveSize" varchar (10) NULL ,
  "MonitorResolution" varchar (20) NULL ,
  CONSTRAINT "PK_ComputerSystem_2__10" PRIMARY KEY  CLUSTERED
  (
    "SSN"
  ),
  CONSTRAINT "FK_ComputerSystem_1__10" FOREIGN KEY
  (
    "SSN"
  ) REFERENCES "dbo"."Student" (
    "SSN"
  )
)
GO
```

```
/****** Object:    Table dbo.HomeContactData      Script Date: 8/27/97 4:26:54 PM
******/
CREATE TABLE "dbo"."HomeContactData" (
   "SSN" varchar (9) NOT NULL ,
   "StreetAddress1" varchar (30) NULL ,
   "StreetAddress2" varchar (30) NULL ,
   "City" varchar (20) NULL ,
   "State" varchar (2) NULL ,
   "Country" varchar (10) NULL ,
   "ZipCode" varchar (9) NULL ,
   "VoiceNumber" varchar (20) NULL ,
   "FaxNumber" varchar (20) NULL ,
   "EmailAddress" varchar (50) NULL ,
   CONSTRAINT "PK_HomeContactData_2__10" PRIMARY KEY  CLUSTERED
   (
     "SSN"
   ),
   CONSTRAINT "FK_HomeContactData_1__10" FOREIGN KEY
   (
     "SSN"
   ) REFERENCES "dbo"."Student" (
     "SSN"
   )
)
GO

/****** Object:    Table dbo.WorkContactData      Script Date: 8/27/97 4:26:54 PM
******/
CREATE TABLE "dbo"."WorkContactData" (
   "SSN" varchar (9) NOT NULL ,
   "Organization" varchar (30) NULL ,
   "StreetAddress1" varchar (30) NULL ,
   "StreetAddress2" varchar (30) NULL ,
   "City" varchar (20) NULL ,
   "State" varchar (20) NULL ,
   "Country" varchar (10) NULL ,
   "ZipCode" varchar (9) NULL ,
   "VoiceNumber" varchar (20) NULL ,
   "FaxNumber" varchar (20) NULL ,
   "EmailAddress" varchar (50) NULL ,
   CONSTRAINT "PK_WorkContactData_2__10" PRIMARY KEY  CLUSTERED
   (
     "SSN"
   ),
   CONSTRAINT "FK_WorkContactData_1__10" FOREIGN KEY
   (
     "SSN"
   ) REFERENCES "dbo"."Student" (
     "SSN"
   )
)
GO
```

The following HTML file was used for testing the **two-tier** design for the one-host test.

Note the parameters for the applet.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage 2.0">
<title>Two-tier Replication Test Applet</title>
</head>

<body>

<h1 align="center">Two-tier Replication Client Applet</h1>

<hr>

<p><applet code="macker/mirror2tier/ClientReplicatorApplet.class" align="middle"
width="600" height="400" name="Two-tier Database Replication Test Applet"
alt="If you had a java-enabled browser, you would see an applet here.">
<param name="useDefaults" value="true">
<param name="destJDBCDriver" value="sun.jdbc.odbc.JdbcOdbcDriver">
<param name="destURL" value="jdbc:odbc:LocalServer1;database=">
<param name="destDSO" value="test2">
<param name="destUID" value="sa">
<param name="destPWD" value="macker">
<param name="sourceJDBCDriver" value="sun.jdbc.odbc.JdbcOdbcDriver">
<param name="sourceURL" value="jdbc:odbc:LocalServer1;database=">
<param name="sourceDSO" value="test1">
<param name="sourceUID" value="sa">
<param name="sourcePWD" value="macker">
Java 1.1 applet not supported by your browser.</applet></p>
</body>
</html>
```

The following HTML file was used for testing the **three-tier** design for the one-host test.

Note the parameters for the applet.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="Microsoft FrontPage 2.0">
<title>Three-tier Replication Test Applet</title>
</head>

<body>

<h1 align="center">Three-tier Replication Client Applet</h1>

<hr>
```

```html
<p><applet code="macker/mirror3tier/ClientReplicatorApplet.class" align="middle"
width="600" height="400" name="Three-tier Database Replication Test Applet"
alt="If you had a java-enabled browser, you would see an applet here.">
<!param name=USE_ORB_LOCATOR value=null>
<param name=ORBgatekeeperIOR value=>
<param name="useDefaults" value="true">
<param name="destJDBCDriver" value="sun.jdbc.odbc.JdbcOdbcDriver">
<param name="destURL" value="jdbc:odbc:LocalServer1;database=">
<param name="destDSO" value="test2">
<param name="destUID" value="sa">
<param name="destPWD" value="macker">
<param name="sourceJDBCDriver" value="sun.jdbc.odbc.JdbcOdbcDriver">
<param name="sourceURL" value="jdbc:odbc:LocalServer1;database=">
<param name="sourceDSO" value="test1">
<param name="sourceUID" value="sa">
<param name="sourcePWD" value="macker">
Java 1.1 applet not supported by your browser.</applet></p>
</body>
</html>
```

Vita

Captain Michael L. Acker was born on 29 April 1964 in Madison, Wisconsin. He graduated from Deerfield High School, Deerfield, Wisconsin in 1982. He enlisted in the Air Force in 1986 as a Scientific Laboratory Technician. In 1989, he was selected as one of the Air Force's 12 Outstanding Airmen of the Year. He was selected for the Airman Education and Commissioning Program (AECP) in 1990 and was reassigned to Wright State University to complete a Bachelor of Science degree in Computer Science. He graduated Suma Cum Laude in 1993. After graduating Officer Training School, his first officer assignment was to the 38th Engineering Installation Wing at Tinker AFB, Oklahoma, where he was program manager for the Standard Automated Remote-to-AUTODIN Host (SARAH) software system and a software systems analyst. He remained in Oklahoma until May of 1996, when he began attending the Air Force Institute of Technology.

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1997	Master's Thesis

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
AN EXAMINATION OF MULTI-TIER DESIGNS FOR LEGACY DATA ACCESS	

6. AUTHOR(S)

Michael L. Acker, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Air Force Institute of Technology, WPAFB OH 45433-7765	AFIT/GCS/ENG/97D-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ AU/AETC/EDXD Jerry A. Boling, Ph.D. 55 LeMay Plaza South Maxwell AFB, AL 36112	

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution unlimited	

13. ABSTRACT (Maximum 200 words)

This work examines the application of Java and the Common Object Request Broker Architecture (CORBA) to support access to remote databases via the Internet. The research applies these software technologies to assist an Air Force distance learning provider in improving the capabilities of its World Wide Web-based correspondence system.

An analysis of the distance learning provider's operation revealed a strong dependency on a non-collocated legacy relational database. This dependency limits the distance learning provider's future web-based capabilities. A recommendation to improve operation by data replication is proposed, and the implementation details are provided for two alternative test systems that support data replication between heterogeneous relational database management systems. The first test system incorporates a two-tier architecture design using Java, and the second system employs a three-tier architecture design using Java and CORBA.

Data on replication times for the two-tier and three-tier designs are presented, revealing a greater performance consistency from the three-tier design over the two-tier design for varying client platforms and communications channels. Discussion of a small-scale proof-of-concept system based on the three-tier design is provided, along with a presentation of the potential for the technologies applied in this system to benefit Air Force web-based distance learning.

14. SUBJECT TERMS		15. NUMBER OF PAGES
Data Base, Data Management, Distributed Data Processing Education, Training, Internet, Computer Programming		152
		16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102